Commodore

Cena 12 tys. zł.
nr indeksu 355275

11-12 '92

KEFIR

Miesięcznik Użytkowników Komputerów C-64 i Amiga



Commodore



nr indeksu 355275

Wydawca:
KEBAB - sp. z o.o.
ul. Wojciechowskiego 28
PL - 71-476 Szczecin
telefon (091) 776-74

Redaguje kolegium
w składzie:
Krzysztof Kobus
Patryk Łogiewa
Grzegorz Mikuła
Krzysztof Moroń
Marcin Orłowski
Zbigniew Piotrowicz
Miłosław Smyk
Paweł Sołtysiński

Redaktor naczelny:
Patryk Łogiewa

Szef działu AMIGA:
Krzysztof Kobus
tel. (091) 525-336

Szef działu C-64:
Paweł Sołtysiński
tel. (091) 776-74

Projekt okładki:
Tomasz Kuczyński

Commodore



PRENUMERATA

Każdy egzemplarz zakupiony bezpośrednio u nas kosztuje odpowiednio:
numery: 1; 2/3; 4; 5; 6 - **9,5 tys. zł**
numery: 7/8; 9; 10 oraz następne - **11 tys. zł.**

Oznacza to, że można zamówić numery zaległe jak też zaprenumerować jeszcze nie wydane. Odbywa się to tylko poprzez dokonanie odpowiedniej wpłaty na nasze konto. Na odwrocie każdego odcinka kuponu wpłaty należy dokładnie napisać, których numerów wpłata dotyczy.

W przypadku prenumeraty można zamawiać numery do końca aktualnego okresu "małej prenumeraty". W roku 1993 "mała prenumerata" będzie obejmować cztery egzemplarze, z podziałem roku na trzy okresy:

I - numery 1, 2, 3, 4 - 4 egz. x 11 tys. = 44 tys. zł
II - numery 5, 6, 7, 8 - 4 egz. x 11 tys. = 44 tys. zł itd.
(jeśli do tego czasu cena nie ulegnie zmianie).

Prosimy nie przysyłać do redakcji dowodów wpłat.

Nasze konto:
Pomorski Bank Kredytowy
II Oddział w Szczecinie
numer konta: 368113-25771-136

Podajcie dokładny adres, imię i nazwisko zamawiającego oraz numery egzemplarzy, których wpłata dotyczy na odwrocie każdego z odcinków blankietu wpłaty.

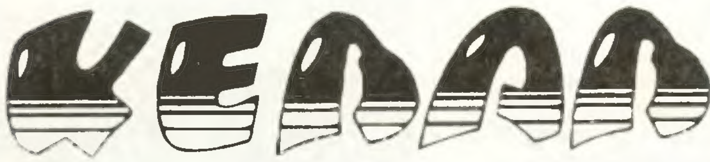
Przy okazji prosimy o podanie dokładnego adresu następujące osoby:
DOBIJA GRZEGORZ
KITA WALDEMAR
KOBASOWICZ WACŁAW

I ostatnia prośba - **wszelką korespondencję, wpłaty etc. kierujcie tylko i wyłącznie na adres redakcji.**

REKLAMA

Ogłoszenia drobne od osób indywidualnych (do 10 słów na kuponie wyciętym z III-ciej strony okładki) przyjmujemy bezpłatnie. Ogłoszenia drobne od osób prawnych oraz zawierające powyżej 10 słów - 1000 zł za słowo.
Ogłoszenia ramkowe (minimalny format 20 cm²):
1 cm² - 4,5 tys. zł, cała strona 2,5 mln. zł,
cała IV strona okładki - 4 mln. zł, 1/2 tej strony - 2,5 mln. zł,
dodatkowy kolor - odpowiednio 50 % drożej.

Ogłoszenia prosimy przesyłać listem poleconym.



Nr 11-12
Listopad-Grudzień 1992

World of Commodore '92
- relacja z targów we Frankfurcie na stronie 2

Co to jest kompresja danych?
strony: 5, 16, 25, 26, 29, 33

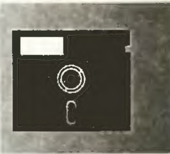
O przerwaniach w asemblerze
- czytaj na stronie 10

Testy trzech rozszerzeń pamięci do A500
- strona 19

Charblaster 2
- cruncher do wklepania na C-64

Spis treści :

















Targi, targi i... po targach! Po raz pierwszy w tym roku odbyła się we Frankfurcie nad Menem impreza targowa o nazwie "World of Commodore", w skrócie "WoC". Impreza odbywała się w dniach 26 do 29 Listopada w hali nr 6 (nie, jak wcześniej zapowiadano, 5) terenów targowych w/w miasta.

W tej wielopiętrowej hali organizatorzy zagospodarowali dwa piętra, z czego jedno (niższe) zostało przeznaczone na tzw. "Entertainment", czyli ogólnie mówiąc wszelką komputerową rozrywkę, a drugie na zastosowania poważniejsze.

Zarówno na jednym jak i na drugim poziomie centralnym punktem było stanowisko firmy Commodore. Można tam było "pomacać" wszystkie nowe produkty z serii Amiga oraz kilka "smutniaków" (czytaj komputerów PC) produkcji tej samej firmy. Najbardziej oblegane były oczywiście A4000, które w ilości ok. 10 sztuk, firma potrafiła wygospodarować na cele prezentacyjne. Na wszelkie pytania typu: Czemu nie można kupić A4000 u dystrybutorów?

Rzecznicy odpowiadali: ..eee.. noo.. boo... aktualnie... Commodore ma trochę problemów z dystrybucją A4000... Kiedy te problemy się skończą? ...eee noo... itp itd. Trudno! Wygląda na to, że jak wreszcie skonstruowano jakąś Amigę, przy pomocy której można poośmieszać wszelkiej maści "pecetowców", to przyjdzie nam czekać i zapisywać się w kolejki społeczne aby móc ją kupić. Co mniej wytrwali potencjalni nabywcy pewnie się zniechęcą i kupią sobie np. jakiś bardziej profesjonalny komputer, choćby AT 286-16...

Co jeszcze można było zobaczyć na stanowiskach Commodore? Choćby zapowiadaną od dawna A1200! Cóż to za dziwo opisaliśmy już w poprzednim numerze, tym razem możemy je nawet pokazać. W przeciwieństwie do A4000, A1200 można było nie tylko pooglądać ale nawet kupić. Z możliwości tej skorzystało sporo osób i bardzo dużo odwiedzających wychodziło niosąc ze sobą pudełko z napisem "Amiga 1200".

Oprócz wymienionych nowości demonstrowano również starsze produkty jak np. Amiga-CDTV, czy A600/HD. Sama impreza, muszę stwierdzić, rozczarowała mnie.

Mimo kilkumiesięcznej promocji we wszystkich pismach komputerowych, rozmiary i rozmach z jakim została przygotowana, nie dorównuje jak chociażby nawet wystawie "Amiga" w Berlinie z jej dawnych, tłustych czasów. Z rozmaitych ciekawostek produktów innych firm, należy wymienić produkty firmy *KCS*.

Ten znany holenderski producent rozmaitych dodatków dla Amigi i C-64 jak zwykle stanął na wysokości zadania. Sztandarowy produkt KCS, jakim jest *Power PC Board*, istnieje już w wersji dla A600. Oprócz tego dla zarejestrowanych użytkowników tego emulatora, firma przygotowała bezpłatną aktualizację oprogramowania zawiadującego PC-Board'em. Aktualna wersja (V4.5) działa na nieco innej zasadzie niż poprzednie, co pozwala na dalsze podniesienie kompatybilności.

Dodatkowo w aktualnej wersji zostały zintegrowane procedury obsługi dźwięku typu "ADLIB" i "SOUNDBLASTER". Oznacza to, że np. gry z pecetów, które wykorzystują powiedzmy kartę SOUNDBLASTER (dla niewtajemniczonych: jest to karta, która umożliwia zwykłemu pecetowi generowanie czegoś trochę mniej nieprzyjemnego dla ucha niż pikanie zainstalowanego wewnątrz głośnika) mogą być również uruchamiane na Power-PC-Board i Amiga wygeneruje odpowiednie dźwięki poprzez swoje kanały Audio.

Firma KCS prezentowała również listę programów i gier, które bez zastrzeżeń działają na emulatorze. Dla ewentualnych niedowiarków przygotowano podłączony do A500+ twardy dysk o pojemności 1200MB (tak, to nie pomyłka: tysiąc dwieście megabajtów) na którym były zainstalowane prawie wszystkie programy wymienione w liście tak aby każdy mógł sobie sprawdzić.

Dodatkowo, w nowej wersji oprogramowania, została przyspieszona jeszcze (!!!) obsługa dysków oraz ekranu. Aktualnie szybkość obsługi dysków elastycznych jest na tyle wysoka, że żaden PC, z którym było mi dane się zetknąć nie osiągał nawet porównywalnej. Inną istotną pozycją ze stoiska KCS, jest przygotowywany aktualnie podwójny napęd dysków 3,5 cala o poczwórnej gęstości zapisu. Jako istotny szczegół należy zanotować, że stacje te mają być kompatybilne ze wszystkimi spotykanymi dotychczas sposobami zapisu HD na dyskietkach 3,5 dla Amigi.

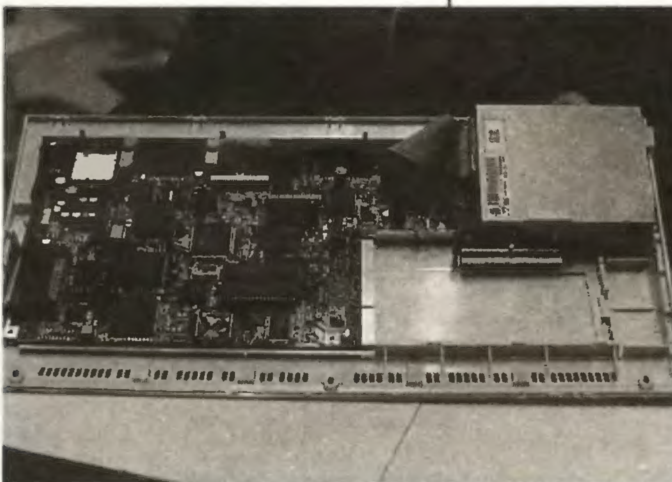


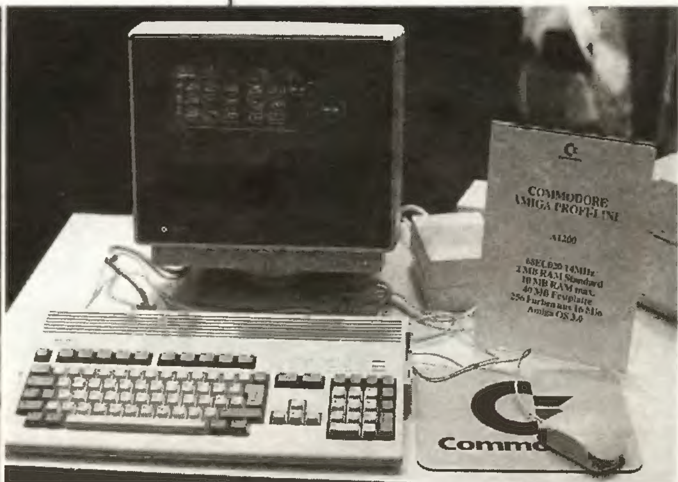
W tym również z tym spotykanym w A3000/4000, czy opisywanym przez nas niedawno napędzie Trinology HD. Przewagą ma tu być ponownie szybkość obsługi. Zgodnie z zapowiedziami, ma ona być od dwukrotnie większa niż w przypadku A4000. No i jeszcze jeden plus. Można ten napęd podłączyć do każdej Amigi, gdyż będzie sprzedawany jako zewnętrzny.

Produkt ma się pojawić w sprzedaży na początku przyszłego roku jak tylko zostaną zakończone prace nad softwarem. Na stanowiskach innych firm, mało było szokujących nowości. Dużym powodzeniem cieszyła się, ze względu na stosunkowo niską cenę, karta *Opal Vision.*

Oprócz tego firma *Masoboshi* przygotowała ciekawostkę - *Real Time Video Digitizer* 24-o bitowy o zaskakująco niskiej cenie. W dziedzinie oprogramowania prawdziwy boom przeżywają programy do obróbki obrazu. Oprócz znanych i uznanych produktów takich jak *ADPro*, czy *ImageMaster* pojawiły się już pakiety innych firm umożliwiające rozmaite operacje na obrazie.

Nadal jednak prym wiedzie firma *ASDG. Morph Plus* to obok ADPro drugi produkt, który prawdopodobnie uplasuje się bardzo wysoko w notowaniach profesjonalistów, oczywiście nie u nas w Polsce. W Polsce nie używa się zabawek do celów profesjonalnych...





Networks

Od zamierzchłych czasów - człowiek był znany ze swego lenistwa i wszędzie, gdzie tylko mógł - ułatwiał sobie życie. Czasami wysługiwał się w tym celu innymi ludźmi, czasami zwierzętami, lecz zarówno w jednym, jak i w drugim przypadku nie zawsze wszystko odbywało się zgodnie z jego planem, bo w końcu któż lubi być wykorzystywany???

Sytuacja uległa jednak zmianie z chwilą wynalezienia komputera. To bezduszne urządzenie okazało się być wyjątkowo uległe i narzucone obowiązki - wykonywało z niebywałą precyzją i dokładnością, a co najważniejsze - wszelkie operacje mogły być przez nie powtarzane setki razy - bez najmniejszych objawów znudzenia, czy zniechęcenia. Czasami tylko w natłoku danych - nawet maszyna ulegała dezorientacji, a wtedy... Użytkownicy AMIGI znają to uczucie, kiedy to ich "przyjaciółka" góruje nad nimi, odmawiając dalszej pracy (lub też raczej... "guruje"). Wkrótce jednak "skrzyneczka" - zwana komputerem - mimo wielu zalet - zaczęła sprawiać pewne kłopoty, kiedy to okazało się, iż wprawdzie określony program jest przez nią wykonywany nawet przez bardzo długi okres czasu, niekiedy jednak ten okres jest stanowczo za długi. Ale i ten problem przedsiębiorczy człowiek rozwiązał, projektując coraz to szybsze jednostki. Jak to jednak w życiu bywa - nigdy nie ma tak dobrze, aby nie mogło być jeszcze lepiej. Przetworzone dane musiały zostać przecież gdzieś przechowane, a możliwość ich udostępnienia innym współpracownikom - stała się wkrótce nieodzowną koniecznością. Pracując na jednostkowym komputerze - przekazanie przetworzonych danych koledze na sąsiednim stanowisku - wiązało się najczęściej z zapisaniem informacji na dyskietce, bądź też skopiowaniem ich z innego miejsca, co z kolei pociągało za sobą konieczność opuszczenia aktualnie używanego programu, następnie trzeba było "przespacerować się" do wybranego stanowiska (często do innego budynku), wręczyć danej osobie dyskietkę - niejednokrotnie po to, aby dowiedzieć się, iż z pewnych bliżej nieokreślonych przyczyn - nie można jej odczytać, po czym zacząć wszystko od początku.

Istotnie, było to kłopotliwe...

Dlaczego twierdzę, iż: "było"??? Ponieważ wszystkiemu można zaradzić, jeśli się tylko chce. Najprostsze rozwiązanie stanowi połączenie obu stanowisk przewodem, co umożliwi transmisję danych bez zbędnych operacji. No tak, oczywiście! Ale co zrobić, gdy tych "końcówek" będzie więcej??? Weźmiemy drugi przewód, kolejny, jeszcze jeden i... plątanina kabli będzie nam przypominać **sieć**!!!

Tak oto powtórnie wpadliśmy na genialny w swej prostocie pomysł, który dodatkowo wcale nie jest nowy. Już pod koniec lat '70, kiedy to twardy dysk stanowił prawdziwego "białego kruka" - było niemożliwością, aby każdy komputer biurowy posiadał to niezmiernie przydatne urządzenie; rozwiązanie nasuwało się samo: należało zakupić szybką jednostkę główną, wyposażoną w HD, a następnie przyłączyć inne "końcówki", mające dostęp do wszelkich peryferii, oferowanych przez komputer centralny.



Tak też uczyniono, zaś wkrótce zaczęły pojawiać się pierwsze systemy sieciowe "na dużą skalę". Nie było w tym zresztą nic dziwnego, gdyż oprócz oszczędności (ze względu na wyeliminowanie z kosztów ogólnych HD dla każdego stanowiska) - dodatkowo umożliwiono użytkownikom dostęp do ogółu informacji. Dla przykładu w 1983 roku firma *Novell* zaprojektowała sieć, zwaną *NetWare*, która to - stale ulepszana - stanowi obecnie jeden z najpopularniejszych systemów, wykorzystywanych w świecie business'u. W tym miejscu pominę może szczegółowe omówienie całej struktury - tak ze względu skali problemu, jak i dlatego, iż na temat "NetWare'u" zapisano już wiele stron. Uważam jednak, iż ogólne wiadomości warto przypomnieć - tym bardziej, iż wielu Czytelników spotyka się z tym zagadnieniem po raz pierwszy.

Będąc podłączonym do sieci - korzystamy ze wszystkich udogodnień, dostępnych w tej sytuacji, jak na przykład: dyski twarde, drukarki, etc. Komputer, na którym pracujemy (zwany stacją roboczą) - może być dodatkowo wyposażony w HD, jednakże nie jest to konieczne; wystarczą zwykłe stacje dysków elastycznych, bo przecież przetwarzane informacje warto czasami zachować także do użytku własnego.

Aby w sieci nie zapanował chaos, potrzebny jest jednak jakiś nadzorca (tak człowiek, jak i maszyna), kierujący pracą całego systemu. Z reguły - jest to względnie szybka jednostka (PC 486), z potężnym twardym dyskiem i pamięcią RAM rzędu 8 [MB], bądź więcej. Maszyna taka zwana jest serwe-

rem (ang: file server) i to właśnie w tym miejscu są przydzielone wszelkie urządzenia zewnętrzne, jak np.: drukarki sieciowe, modemy, dodatkowe pamięci zewnętrzne, etc. Serwer może służyć jedynie do sterowania siecią, jak również istnieje możliwość wykorzystania go, jako stacji roboczej.

Narzucone zadania wykonuje aktualnie wykorzystywana stacja robocza, sieć umożliwia jedynie komunikowanie się z innymi stanowiskami, a także w pewnym zakresie - pozwala na wspólną pracę nad tym samym zbiorem, oraz wspólne wykorzystywanie dostępnych urządzeń. System *Novell'a* - należy raczej do grupy sieci lokalnych (tzw.: LAN - Local Area Network), jednakże istnieje możliwość podłączenia się do innej sieci, aby w miarę osiągania kolejnych szczebli - osiągnąć strukturę, zwaną siecią globalną (WAN).

Korzyści z istnienia takiej "siatki" są znaczne, jednakże należy pamiętać, iż oprócz odpowiedniego przewodu połączeniowego - potrzebujemy jeszcze stosowną kartę, umożliwiającą wykorzystanie danej drogi komunikacji, a przede wszystkim odpowiedni program. Spośród kart sieciowych - najbardziej popularne są obecnie: ARCNET (2.5 [Mbit/s], 600m - odległość węzłów), D-LINK (1 [Mbit/s]), ETHERNET (10 [Mbit/s], 500m - odległość węzłów); uzyskanie większych zasięgów jest oczywiście możliwe, lecz wiąże się z dodatkowymi nakładami finansowymi..

Pozostaje jeszcze udzielić odpowiedzi na pytanie: co przemawia na korzyść sieci *Novell'a*??? Dlaczego są one tak popularne? Niewątpliwie dużą zaletę stanowi bezpieczeństwo danych. Specjalne systemy zabezpieczają informacje przed zapisaniem na uszkodzonym fragmencie dysku, inne są odpowiedzialne za to, by zdarzenia losowe nie spowodowały utraty choćby części informacji (z reguły stosuje się tzw. mirroring, tj. jednoczesny zapis na dwa dyski, by w razie awarii jednego z nich - nie przerywać pracy sieci, tylko dokonać odpowiedniej naprawy w trakcie jej działania); istnieją oczywiście także inne sposoby ochrony informacji (gdyż jak nietrudno zauważyć - w.w. metoda nie jest oszczędna, jeśli chodzi o fizyczne wykorzystanie nośników); istnieje też możliwość zasilania całości z akumulatorów w razie zaniku napięcia zasilania (ang.: UPS - Uninterruptable Power Suply), jak również wyposażenia całości systemu w szereg przydatnych elementów.

Na uwagę zasługują także tzw. dyski poszukiwań. W momencie, w któryn zdecydujemy się na uruchomienie danego programu - musimy podać jego nazwę. Komputer sprawdza więc aktualny katalog, w którym właśnie pracujemy, lecz przecież nie jest możliwe, abyśmy posiadali wszelkie dostępne zbiory, dlatego też w normalnym trybie - otrzymalibyśmy komunikat o błędzie; jeśli jednak zdefiniowano dyski poszukiwań - komputer tam właśnie się skieruje, by odnaleźć interesujący nas plik, co oczywiście zwiększa komfort pracy.

Jedną z ważniejszych kwestii - jest również ochrona osobistych danych przed niepowołanymi osobami. Każdy z użytkowników sieci ma w 100% dostęp do swoich danych, ograniczony (zależnie od uprawnień) - do zbiorów "globalnych", zaś w ogóle nie ma dostępu do zbiorów innych osób, korzystających z sieci. Wyklucza się więc możliwość uszkodzenia informacji przez przypadek, bądź specjalnie, zmniejszając jednocześnie prawdopodobieństwo ich kradzieży do minimum. Dostęp do nich ma tylko określony użytkownik, wyróżniający się niepowtarzalnym imieniem sieciowym, mogący dodatkowo zabezpieczyć dostęp do informacji odpowiednim hasłem.

Jeden z użytkowników sieci znajduje się na pozycji uprzywilejowanej i określany jest mianem administratora (ang.: Supervisor). On jeden ma władzę nieograniczoną, wolno mu zezwalać na pracę w sieci innym użytkownikom, a pracującym aktualnie - odbierać ten przywilej. To on sprawuje bezpośrednią władzę nad systemem i jest odpowiedzialny za jego poprawną pracę. Zwykły użytkownik posiada "okrojone" prawa, tzn. administrator może mu zezwolić np. jedynie na odczyt pewnych danych (bez możliwości ich zmieniania, bądź kasowania), lub też zabronić mu dostępu do określonych podkatalogów; pod tym względem użytkownik jest całkowicie zależny od woli administratora i bez jego zezwolenia - nie jest w stanie nic uczynić.

Te pozorne ograniczenia - normalnie nie są jednak uciążliwe dla korzystających z sieci i umożliwiają efektywną pracę. Administrator może również zezwolić na dostęp użytkownika do danych jedynie w określonych dniach i godzinach, może określić maksymalną liczbę odwołań do systemu (tzw. logowań), sposób dostępu do danych (tj. odczyt/modyfikacja), możliwość tworzenia nowych zbiorów i podkatalogów, oraz ich kasowanie.

Może także nadać prawo zmiany atrybutów podkatalogów, czy też zbiorów, zmiany ich nazw (bez zmiany ich zawartości), jak również prawa modyfikacji wszystkich praw, za wyjątkiem statusu nadzorcy. Na temat innych opcji oferowanych przez system - można by jeszcze napisać wiele, lecz już w.w. - świadczą o dużym bezpieczeństwie informacji, znajdujących się w sieci.

W tym miejscu Czytelnik może zapytać: "No dobrze, ale cóż ma wspólnego *Novell* z AMIGĄ lub C64? Przecież nasze czasopismo jest poświęcone właśnie tym komputerom!"

Rzeczywiście, *Novell* nie widział nigdy potrzeby umożliwienia pracy w sieci tak obecnie popularnym komputerom, jakie stanowią Amigi (wiadomo: wszak to tylko zabawka - przyp. red.). Czy jednak tak już miało pozostać na zawsze??? Na pewno nie! Podobnego zdania byli też przedstawiciele firmy *Oxxi*, tworząc Amiga Client Software (ACS). ACS umożliwia na podłączenie do sieci *Novell'a* do 250 Amig; jednakże - oprócz oprogramowania - potrzeba coś jeszcze: wymagana jest mianowicie karta ETHERNET, łącząca A1500/2000/3000 fizycznie z siecią, za pomocą

przewodów, dochodzących do karty i z niej wyprowadzonych.

Tak więc koszt stworzenia takiej sieci - nie jest wysoki, jednakże serwer powinna stanowić jednostka zdecydowanie szybsza, np. 486 PC, lub też coś innego, pracującego pod kontrolą systemu Unix.

System sieciowy został tak pomyślany, aby każdy komputer, pracujący w sieci - miał dostęp do tych samych informacji, poprzez ten sam serwer, niezależnie od tego, czy będzie to Mac, PC, czy Amiga.

Możliwość taka istnieje w związku z tym, iż cały system rządzi się własnymi prawami i podstawowe dane przechowywane są w pod kilkoma różnymi postaciami - stąd też Mac otrzyma informacje w żądanym formacie, dla PC - zostanie zachowana konwencja ośmiu znaków przypadających na nazwę pliku, zaś dla Amigi - nie zabraknie zbioru '.info'. W ten sposób zachowana została harmonia i nikt nie może czuć się pokrzywdzony.

A co można powiedzieć na temat pracy Amigi w sieci? Oczywiście wszystkie wcześniej wymienione udogodnienia - tyczą się jej także, zaś Gary Fenton miał okazję "pobawić się" takim systemem; wczytanie DPaint'a IV - odbyło się w ułamku sekundy. Poprawnie przebiegały również operacje wysyłania komunikatów do innych użytkowników sieci. W tym miejscu należy wspomnieć o drobnej zalecie, jaka wiąże się z multitaskingiem.

ACS zawiadamia użytkownika o ewentualnym nadejściu wiadomości, przy czym zostanie ona zachowana do momentu, gdy korzystający ze stacji roboczej - nie odbierze jej. I cóż w tym dziwnego? - może ktoś zapytać.

Prawdę mówiąc - nic; tylko, że popularny PC wszelkie komunikaty "wyrzuca" bezpośrednio na ekran, co po pierwsze może być denerwujące dla użytkownika, który musi przerywać pracę, by usunąć zbędną wiadomość z pola widzenia, po drugie łatwo przewidzieć, w jakiej formie odbierzemy komunikat, gdy będziemy pracować w trybie graficznym.

Programiści będą więc musieli jeszcze trochę popracować nad tym - co system Amigi oferuje już od dawna... Inną ciekawostką jest wykorzystanie około tysiąca Commodore'owskiego SL386sx PC w blisko 211 biurach podróży.

Gary Fenton zastanawiał się, dlaczego nie zainstalowano w nich komputerów serii AMIGA? Czyżby obawianoo się, iż personel cały czas zajmowałby się grami, zamiast rezerwacją lotów???

Być może sytuacja ta ulegnie zmianie w niedalekiej przyszłości, lecz jak na razie - jedyny komentarz, tyczący się "złotej myśli" Gary'ego może być tylko jeden: *Magnum in parvo...*

Krzysztof Franckowski

P.S.W artykule wykorzystano materiały z "Novell Networks" (AUI, Jul. '92)

Co to jest kompresja danych ?

Bardzo często używamy różnego rodzaju programów służących do zmniejszania wielkości innych programów lub danych tak, aby zajmowały one mniej miejsca na dysku lub w pamięci komputera, także aby skrócić czas transmisji danych pomiędzy urządzeniami - np. modemami. I tak, korzystając z popularnych cruncher'ów, bądź archiwizerów od czasu do czasu doznajemy szoku widząc jak z kilkusetkilobajtowego zbioru danych pozostaje dwie-trzecie, a czasem nawet połowa lub mniej. I w każdej ciekawskiej głowie powstaje pytanie, jak się to dzieje, że ze stu bajtów robi się siedemdziesiąt. Czyżby dla otrzymania zysku na pamięci usuwano jakąś niepotrzebną informację ? Poniekąd tak. Przykładowo - który fragment pliku z tekstem tego artykułu jest zbyteczny? Abstrahując od wszystkich złośliwych odpowiedzi na zadane pytanie, trudno znaleźć jakiś sposób na zredukowanie tekstu i następnie na jego odtworzenie. Istotne jest tutaj także owo odtworzenie tekstu, ponieważ nie jest naszym celem w przypadku komresji danych jedynie redukcja zajętości pamięci.

Każdy spakowany zbiór powinien dać się rozpakować do źródłowej postaci - jeżeli pominiemy ten mały szczegół, sprawa się upraszcza i każdą ilośc danych natychmiast potrafimy zredukować z dowolnym zyskiem z przedziału od stu do zera procent i to na setki sposobów. Tak więc tekst należy w sensowny sposób zakodować tak, aby zajmował mniej miejsca.

Tylko w jaki sposób ? Jest wiele metod, od prymitywnych do bardzo skomplikowanych. Najprostszą jest metoda "znacznikowa". Polega ona na znalezieniu wszystkich powtarzających się znaków podczas przeglądania tekstu. Przykładowo ciąg znaków " aaaaaaaaaaaaaaa   bobo" możemy zakodować w następujący sposób: jeden znak " ", piętnaście znaków "a", pięć " " oraz kolejno bajty niepowtarzalne "bobo". Zakodowany w ten sposób tekst zawiera, pomiędzy niepowtarzającymi się sekwen-cjami, rozkazy wypełniania pamięci określoną wartością. Dla zobrazowania można przedstawić przykład kodów rozkazów oraz danych takiego "znacznikowca".

Bajt rozkazu w formacie nooooooo, gdzie bit n oznacza jeden z dwóch możliwych rozkazów (n=1 - dane niepowtarzalne, n=0 - powtarzalne), siedem bitów ooooooo zawiera licznik dla tego rozkazu. Powyższy ciąg znaków możemy w ten sposób zakodować uzyskując

1:$81, 2:" ", 3:$0f, 4:"a", 5:$05, 6:" ", 7:$84, 8:"bobo", 12:$00

1: $81 = %10000001 ,

a więc jeden niepowtarzalny bajt o kodzie

2:"(spacja)"

3: $0f = %00001111 ,

piętnaście powtarzalnych bajtów o kodzie

4:"a"

5: $05 = %00000101 , pięć razy

6:"(spacja)"

7: $84 = %10000100 ,

cztery kolejne niepowtarzalne bajty "bobo".

Sumując, w jedenastu bajtach zakodowaliśmy źródłowe 25 bajtów tekstu, należy tu dodać oczywiście jeszcze jeden rozkaz na końcu o np. zerowym parametrze aby poinformować program dekodujący o końcu zakodowanego ciągu.

12: $00

Daje to nam 52 procenty zysku. Ta prosta metoda znacznikowa znajduje zastosowanie w przypadkach zbiorów danych o prostej strukturze, np. prymitywna grafika, czy tablice znaczników w bazach danych. Stosowana jest również do wstępnego przygotowania danych przed zastosowaniem innych metod kompresji.

Najefektywniejszym sposobem kompresji jest metoda wyszukiwania powtarzających się sekwencji bajtów - *Crunching*. Najogólniej można wyjaśnić to na przykładzie tekstu tego artykułu. Przeglądając go, można napotkać wielokrotnie wyrażenia : kompresji, więc, bajt, kod, itp. Powtarzające się wyrażenia są właśnie sekwencjami, które teraz pozostaje tylko zakodować. Sposobów kodowania tą metodą jest wiele, ponieważ ewoluowała ona kilka lat.

Trzonem działania cruncher'a jest program, który przeszukuje pamięć od początku do aktualnego wskazania pamięci, które to zwiększane jest od początku obszaru danych do końca o jeden, lub w przypadku odnalezienia sekwencji - o jej długość. Wskazanie wstecz, nazywane offsetem, jest podawane w postaci adresu względnego odejmowanego od aktualnej pozycji adresowej w zbiorze danych. Można powiedzieć, że offset "ślizga" się za wskazaniem w pamięci.

Tak więc, spakowany kod składa się z przeplatanych rozkazów z danymi niepowtarzalnymi i wskazaniami wstecz w pamięci na sekwencje do skopiowania. Każdy z rozkazów zawiera także długość danych jakich dotyczy.

Przykładowo rozkaz w postaci sekwencji bitów:

nooooooo oznacza dla n=1 dane niepowtarzalne o długości ooooooo,

nooooppp pppppppp oznacza dla n=0 sekwencję o długości oooo ze wskazaniem offsetowym wstecz o wartości ppppppppp co daje możliwość zakodowania jednorazowo danych niepowtarzalnych o długości do 128 bajtów oraz sekwencji o długości do 16 bajtów z przedziału adresowego $800 wstecz.

Oczywiście ten przykład ma tylko zasygnalizować sposób i nie jest jedyną polecaną przez autora konfiguracją parametrów. Optymalnym zakresem offsetu, jak stwierdzono doświadczalnie, jest przedział od $800 do $2000 bajtów. Jeżeli chodzi o długości sekwencji, to najczęstsze są te najkrótsze, czyli 2,3,4-ro bajtowe, itd. Oczywiście długości rozkazów w bitach należy dostosować do wymagań crunchera. Praca tak opisanego procesu jest czasochłonna, program wielokrotnie przeszukuje cały wyznaczony obszar pamięci.

Aby przyspieszyć jego działanie, należy stworzyć w odrębnym miejscu w pamięci słownik, do którego wpisywane będą adresy już odnalezionych sekwencji. Program szukający sekwencji w pierwszej kolejności sprawdza wśród już znalezionych, a dopiero potem ewentualnie przeszukuje obszar objęty offsetem. Oczywiście słownik jest w czasie trwania procesu bez przerwy uaktualniany, co oznacza zarówno dopisywanie nowych adresów jak i zwalnianie pozycji z adresami sięgającymi poza offset.

Jeżeli dysponujemy dużym obszarem dodatkowej pamięci, to aby jeszcze przyspieszyć działanie crunchera, słownik sekwencji stworzyć można w postaci drzewa binarnego (sortowanego wg rosnących wartości). Sposób ten jest ciekawy, lecz warto tu wspomnieć, iż nie jest stosowany ze względu na istnienie skuteczniejszej metody przyspieszenia pracy.

Otóż należy stworzyć tablice adresów pod którymi można znaleźć bajty o wszystkich kolejnych wartościach. Oznacza to, że adresy wszystkich bajtów w obrębie offsetu są pogrupowane w tablicach, gdzie każda tablica zawiera adresy bajtów o takiej samej wartości. Mając bajt o wartości 2 i szukając sekwencji od niego rozpoczynającej się, będziemy korzystać tylko z adresów zawartych w tablicy zawierającej wszystkie adresy dwójek. Pamiętać oczywiście należy o usuwaniu adresów sięgających poza offset. Statystycznie rzecz ujmując, jeżeli przy przeszukiwaniu offsetu o wielkości 4 KB zastosujemy opisany wyżej słownik to, zakładając, że każda wartość bajtu występuje z jednakową częstością, uzyskujemy (4096/256=16) konieczność szukania sekwencji od miejsc na które wskazuje tylko szesnaście adresów w słowniku.

Czyli zamiast przeszukującej pętli powtarzającej się 4096 razy, potrzebujemy tylko szesnastu jej powtórzeń. Oczywiście taka sytuacja jest idealna, normalnie występowanie wartości nie jest równomierne i powtórzeń może być więcej od szesnastu, ale dla innych wartości automatycznie się ta liczba zmniejsza. W ekstremalnym przypadku kiedy jakaś wartość wystąpiła po raz pierwszy to oczywiście jej adres jest wstawiany do słownika, lecz uprzednio nie przeszukujemy pamięci ponieważ w słowniku pod tą pozycją nie było żadnego adresu. Kiedy napotkamy po raz drugi tą samą wartość to sprawdzamy zaistnienie

PRESENTS CRUNCHERS
KRREEK
PRAWIE AUTOMATYCZNY
BLAM! BLAM!
SEMI-AUTOMATYCZNY
KRREEK...
MANUALNY
JOKESTICK

sekwencji tylko raz i dopisujemy nowy adres to słownika, kolejnym razem przeszukujemy trzy razy itd. Przy zastosowaniu tej metody program przyspiesza kilkudziesięciokrotnie. Sposób zorganizowania takiego słownika nie jest już taki prosty ze względu na ograniczoną ilość pamięci. Idealną ilością pamięci potrzebną do alokacji dla słownika offsetu o wielkości $1000 byłby obszar $1000 * 4 * 256, czyli 256 tablic z adresami 4-ro bajtowymi, a w każdej tablicy potencjalnie może się znaleźć $1000 adresów.

Całość wymaga 4 mega bajtów. Autor artykułu może zapewnić, iż dla offsetu wielkości $1000 na słownik wystarczy $2200 bajtów ponieważ jest autorem takiej metody. W celu pominięcia nieoszczędnego kodowania rozkazów za pomocą bitów, zastosowano ciekawy sposób. Z całego kodowanego obszaru wybiera się najrzadziej występującą wartość i następnie używa jako rozkazu. Decruncher po napotkaniu tej wartości traktuje ją jako daną, jeżeli zaraz po niej wystąpi np. zero - w przypadku innej wartości - jako rozkaz z odpowiednim parametrem. Dzięki temu nie ma potrzeby oznaczania rozkazem bloków danych nie objętych kompresją, z tym, że jedna wartośc bajtu będzie zawsze reprezentowana za pomocą dwóch bajtów.

Na zupełnie innej zasadzie opierają się metody kodowania względnego - *Relative Encoding.*

Metoda telemetryczna - *Telemetry Encoding* - koduje względne zmiany wartości pomiędzy kolejnymi słowami ciągu danych (np. bajtami). Na spakowane dane składają się rozkazy ustalające bazową wartość oraz kolejne wartości, które należy do niej dodawać. Ciąg ($40 $43 $46 $48 $3e $40 $40 $80 $79) można zakodować, przy założeniu 5-cio bitowych kodów dodawania i kodu %10000 jako znacznika dla przyjęcia nowej wartości bazowej, w następujący sposób: ($40, %00011 = +3, %00110 = +6, %01000 = +8, %10010 = -2, %00000 = +0, %10000, $80, %10111 = -7)

Z 9 bajtów uzyskaliśmy 6.4 bajta, czyli zyskaliśmy 30 % oszczędności pamięci.

Możemy także przyjąć inną zasadę kodowania, w której nie będzie stałej bazowej wartości, lecz będzie ona uaktualniana w czasie dodawania kolejnych wartości, co powiększy zakres wartości możliwych do zakodowania bez potrzeby ustalania nowej wartości bazowej. Ciąg (0 1 3 5 8 9 $b $10 $16 $1d $20 $20 $1e $17 $11 5 1) przy założeniu 4-ro bitowych kodów: (0 %0001 %0010 %0010 %0011 %0001 %0010 %0100 %0110 %0111 %0011 %0000 %1010 %1111 %1110 %1000 5 %1100)

Z 17 bajtów uzyskaliśmy 10 B, czyli mamy 41% zysku. Metoda ta znajduje zastosowanie przy kompresji ciągów z wynikami pomiarów, gdzie małe różnice pomiędzy kolejnymi wartościami można przedstawić w skróconej postaci. Wyobraźmy sobie jak duże zyski można osiągać przy kompresji danych zapisanych w postaci zmiennoprzecinkowej *Floating Point.*

Jeżeli notacja FP wymaga 12 bajtów, a my stworzymy notację skróconą do 6 bajtów i zastosujemy do zapisywania wartości względnych, to możliwy zysk nasuwa się sam. Technika kodowania telemetrycznego może znaleźć zastosowanie u każdego programisty, ponieważ nadaje się do kompresji różnego rodzaju sinusoid, trajektorii, itp.

Drugą techniką kodowania względnego jest metoda *Digital Facsimile*, stosowana głównie do kompresji grafiki uzyskiwanej ze scanera. Metoda ta wykorzystuje fakt, iż często pomiędzy kolejnymi poziomymi liniami obrazu występują tylko nieliczne zmiany. Jeżeli w odniesieniu do bazowej (górnej) linii wyselekcjonujemy zmiany w kolejnej linii, to zredukujemy dane o jakąś ilość niepotrzebnej informacji. Przykładowo jeżeli N-ta linia ma postać a N+1 to względne zmiany mają postać :

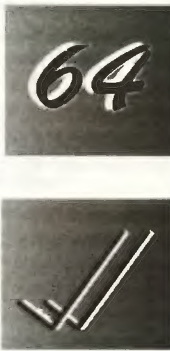
```
...0000001111100111010110001....
...0000011111000010010110011...,
Exor #........1....1..1.1.......1.....
```

Zmiany można zakodować poda jąc bezwzględnie ich pozycje w każdej linii lub względnie podając odległości pomiędzy samymi zmianami. Koniec linii oznaczyć można np. wartościa poza zakresem. Innym sposobem kodowania tych zmian może być mapa bitowa (ustawiony bit oznaczałby zmianę), a sposób kompresji tej mapy byłby już dowolny, metodą znacznikową lub sekwencyjną. Oczywiście długość każdej zakodowanej linii musi być porównana z jej długością oryginalną, a wybrana powinna zostać ta krótsza.

Kolejnymi technikami kompresji danych są metody probabilistyczne, są to kolejno : *Diatomic, Expanding*, kodowanie *Huffmana* i metoda *Shannon-Fano.*

Metoda *Diatomic* polega na przypisaniu jednobajtowym kodom określone statystycznie pary bajtów. Ogólny proces *Diatomic* określić można relacją : znak N+1 + znak N ===> znak specjalny. *Diatomic* nadaje się np. do wstępnej kompresji tekstów, umożliwiając osiągnięcie maksymalnego teoretycznego zysku około pięćdziesięciu procent. Zakodować możemy 255 par, jednak pozostałe (256^2-255) pary muszą pozostać nie zakodowane, więc zastosowanie tej metody musi zostać zawsze uprzednio sprawdzone pod względem opłacalności ze względu na wymagany specyficzny typ danych (np. poprzez kompresję mniejszej partii danych do NIL i sprawdzenie wyniku). Spakowany tą metodą tekst składa się z kodów jednobajtowych od 0 do 254, oznaczających określoną uprzednio parę znaków oraz z kodów dwubajtowych o wartościach kolejno 255 i bajtu nie objętego kompresją.

Praktycznie metoda ta dla tekstów w języku angielskim przynosi średnio 20 % zysku, co w porównaniu z innymi technikami kompresji stawia ją na odległym miejscu. Jednak w połączeniu z metodą znacznikową oraz kodowaniem połówek bajtów, metoda *Diatomic* daje dobre zyski przy tekstach, a połączenie tych trzech metod nazwano *Shrinking*. Przykładowa struktura rozkazów metody *Shrinking* może wyglądać jak następuje. Kody od 0 do $7f zawierają zwykłe kody znaków ASCII. Pozostałym kodom z zakresu $80 do $ff przypisujemy określone statystycznie pary znaków. Jeden z kodów o najwyższych wartościach rezerwujemy sobie na bajty dodatkowe, nie przewidziane przez dotychczasowe kody (po takim rozkazie będzie następowała sekwencja 8 bitów z dowolną wartością). Całość możemy zakodować połówkami bajtów, osobno kodując czy starsze i młodsze 4 bity zmieniają się w stosunku do poprzedzającego znaku. Można również przyjąć kodowanie starszych 3 bitów i młodszych 5 bitów, jak nam wygodniej, a raczej - jak korzystniej.



Metoda *Expanding* wykorzystuje dwa algorytmy, tzn. kompresja dokonuje się w dwóch przejściach. W pierwszej kolejności cały pakowany blok jest kodowany metodą sekwencyjną, a następnie metodą probabilistyczną. Idea probabilistycznej metody *Expanding* opiera się na wykryciu zależności zachodzących pomiędzy kolejnymi bajtami ciągu danych. Ponieważ pakowaniu podlega zazwyczaj konkretna informacja, to istnieje duże prawdopodobieństwo, że pewne wartości mogą występować częściej jedna po drugiej od innych.

Przykładowo w języku polskim większe jest prawdopodobieństwo wystąpienia litery "k" zaraz po "o", niż "z" po "o", natomiast prawdopodpbieństwo, że "y" wystąpi po "o" jest małe. (uwaga: dane wyssane z palca). Metoda *Expanding* w prosty sposób systematyzuje tego typu zależności i jeżeli pakowane dane nie są tzw. szumem, gdzie prawdopodobieństwa powtarzania się dla wszystkich bajtów są zbliżone, można otrzymać zysk kilkudziesięciu procent.

Do metody *Expanding* potrzebujemy dodatkowego bufora na tablice w których program zapisuje informację, której się

"uczy" w czasie kompresji danych. Należy stworzyć tablicę tablic powtarzających się bajtów S(j), gdzie j=0..255, czyli dla wszystkich możliwych wartości bajtu. Każda tablica S(j) zawiera np. 32 wartości (odpowiednio do liczby B(N()) opisanej niżej). czyli S(j)[0..m],gdzie m=0..31(patrz B(N())). Kolejna tablica N(j) zawiera ilości zapamiętanych wartości w zbiorach S(j). Tak więc j oznacza rozważaną wartość, zbiór S(j) zawiera wartości które występują zaraz po j, a N(j) wielkość S(j). Zbiory ostatecznie mają postać {N(j), S(j)[0], ... ,S(j)[N(j)-1] } N(j) może mieć wartość zero jeżeli zbiór S(j) jest pusty.

Algorytm przedstawia przykład pierwszej fazy dekompresji *Expanding* (proces odwrotny do kompresji jest częściej stosowany do opisów ze względu na większą przejrzystość).

WE: oznacza strumień danych wejściowych (spakowanych statystycznie)

WY: oznacza strumień danych wyjściowych (spakowanych sekwencyjnie)

L-C: oznacza ostatnią wartość (poprzedni bajt) m: zmienna

B(N(j)): oznacza maksymalną ilość bitów potrzebnych do zakodowania największej wartości N(j)-1

początek:
{L-C:=0
powtarzaj pętlę do końca WE
{jeśli zbiór S(L-C) jest pusty to
{czytaj 8 bitów z WE i kopiuj wartość do WY}
jeśli (S-C) nie jest pusty to
{czytaj 1 bit z WE
jeśli bit=1 to
{czytaj 8 bitów z WE i kopiuj wartość do WY }
jeśli bit=0 to
{czytaj B(N(L-C)) bitów z WE i nadaj tą wartość zmiennej m kopiuj wartość z S(L-C)[m] do WY }}
nadaj zmiennej L-C wartość ostatniej wartości wysłanej do WY }}
koniec.

W ten sposób możemy regularnie powtarzające się bajty zakodować za pomocą np. pięciu bitów. Niestety nie powtarzające się, bądź te które nie zmieszczą się w tablicach musimy zapisać z pomocą dziewięciu bitów.

Na wynikowo skompresowane dane składają się strumień wyjściowy WY oraz tablice N(0..255) o elementach długości B(N()) bitów i J(0..255,0..k), gdzie k jest maksymalną wartością zapisywalną w N(). Najczęściej stosuje się długości 5 i 6 bitów, czyli 32 i 64 zapamiętywalne wartości powtórzeń dla każdego bajtu.

Następnym etapem dekompresji *Expanding* jest rozkodowanie bloku danych do źródłowej postaci z pomocą algorytmu sekwencyjnego.

Aktualnie WE oznacza WY z poprzedniego algorytmu oraz: NWW: najrzadziej występująca wartość (przykładowo 144),

L(x): zwraca młodsze L bitów z x,

F(x): zwraca 2 jeśli L(x) ma największą wartość (np. dla L=7 F(x)=2 jeśli L(x)=%01111111) , w innym wypadku zwraca 3,

D(x,y): zwraca (starsze (8-L) bitów z x) * $100 + y + 1,

V,C: zmienne,

Dlg: zmienna oznaczająca długość sekwencji.

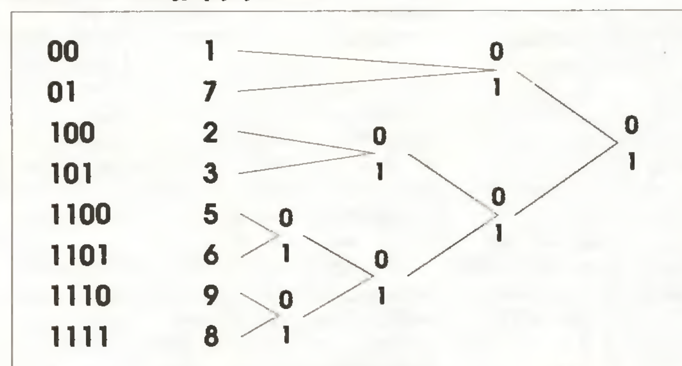
Przed kompresją podaje się programowi parametr określający L. Współczynnik kompresji (Compression Factor) równy 1 => ustala L=7, 2 => L=6, 3 => L=5, 4 => L=4, co daje użytkownikowi pewną dowolność w wyborze zakresów rozkazów jakimi posługuje się program kodujący,

a zatem dla współczynnika kompresji od 1 do 4 uzyskujemy maksymalny offset z zakresu od $200 do $1000 i maksymalną długość sekwencji od $80+$ff do $f+$ff.

początek:
{pprog:=0
powtarzaj pętlę do końca WE
{czytaj 8 bitów z WE i nadaj tą wartość C
case pprog of :
0:{jeśli C nie jest równe NWW to kopiuj C do WY jeśli C=NWW to pprog:=1}
1: {jeśli C=0 to {kopiuj NWW(* =144 *) do WY pprog:=0}
jeśli c<>0 to {V:=C Dlg:=L(V) pprog:= F(Dlg)}}
2: {Dlg:=Dlg+C pprog:=3}
3: {cofnij się wstecz w WY o D(V,C) bajtów i kopiuj Dlg+3 bajtów z tego miejsca do WY (* jeśli ta pozycja jest przed początkiem WY to przyjmuje się, że obszar ten jest wypełniony zerami *)
pprog:=0 }
koniec case}}
koniec.

Kolejnym rodzajem kompresji probabilistycznej jest kodowanie metodą *Huffmana*. Sposób ten opiera się na kodowaniu częściej powtarzających się bajtów za pomocą mniejszej ilości bitów. We wstępnej fazie oblicza się częstości występowania wartości bajtów, co jest równoważne obliczeniu prawdopodobieństw ich napotkania. Następnie tworzy się drzewo binarne, po którego gałęziach poruszać będzie się program kodując dane źródłowe. Załóżmy dowolny ciąg bajtów: {1 2 1 1 2 1 2 3 1 3 1 3 2 1 7 1 7 5 6 7 1 7 1 7 2 7 7 9 8} Częstość występowania wynosi kolejno : 1-10, 2-5, 3-3, 5-1, 6-1, 7-7, 9-1, 8-1 .

Sortując wg malejących wartości powtórzeń : 1-10, 7-7, 2-5, 3-3, 5-1, 6-1, 9-1, 8-1 . Na podstawie uzyskanych wyników drzewo ma następującą postać:



Podany wyżej ciąg możemy zakodować : {1-00 2-100 1-00 1-00 2-100 1-00 2-100 3-101 1-00 3-101 1-00 3-101 2-100 1-00 7-01 1-00 7-01 5-1100 6-1101 7-01 1-00 7-01 1-00 7-01 2-100 7-01 7-01 9-1110 8-1111} czyli: {00100000 01000010 01010010 10010110 00001000 11100110 10100010 00110001 0111101111 } co daje nam 74 bity, a więc z 29 uzyskaliśmy 9,25 bajta, czyli mamy 68% zysku.

Przykład ten jest ekstremalnie uproszczony, ale widać tu zastosowanie tej metody do danych "przemieszanych", nie wykazujących regularności w układzie przestrzennym. Nasze drzewo oczywiście może się rozrosnąć do większych rozmiarów, z tym, że dla każdych dodatkowych dwóch bajtów musimy zużyć jeden bit więcej, tak aby móc je zakodować.

Należy tu przypomnieć, iż jak w każdej metodzie probabilistycznej, wymagane jest jak największe zróżnicowanie częstości wystąpień wartości bajtu. Im bardziej prawdopodobieństwa wystąpień wszystkich wartości bajtu będa zbliżały się do jednej średniej wartości, tym bardziej narażamy się na straty, aż po nieopłacalność stosowania metody probabilistycznej do tego typu danych.

Przy dużej ilości elementów w drzewie spotykamy się z kodami liczącymi wiele bitów. Aby zapobiec takiemu marnotrawstwu, można jeden z liści drzewa poświęcić na oznaczenie nie zakodowanego bajtu, który przy dekompesji byłby pobrany w postaci 8 bitów ze strumienia wejściowego. Ten manewr pozwoli nam na uniknięcie kodowania rzadko spotykanych wartości za pomocą np. trzydziestu bitów, przykład podobnej operacji jest zawarty poniżej.

Zbliżoną do metody *Huffmana* jest metoda kodowania *Shannon-Fano*.

Tutaj również należy w pierwszej kolejności określić prawdopodobieństwa wystąpień elementów, a następnie zbudować drzewo binarne. Elementy ustawiamy w kolejności malejących prawdopodobieństw, następnie grupujemy je parami. Każda para jest w kolejnych kolumnach pionowych oznaczana za pomocą jedynek, a następnie dzielona na dwa przy pomocy jedynki i zera.

11
10
011
010
0011
0010
itd.

W ten sposób możemy zakodować na małej ilości bitów kilka wartości. Ale co się stanie z np. czterdziestą wartością. Do jej zapisania potrzebujemy aż 22 bity, czyli prawie cztery bajty. Stosując pewien trik możemy tę niedogodność usunąć. Zrezygnujemy z oznaczania jednej wartości kodem 010 i odtąd poprzedzać on będzie sekwencję np. 6 bitów, na których zapisać można 64 wartości. Aktualnie kody wyglądają następująco :

11
10
011
0011
0010
00011

00010
000011
000010
0000011 itd.
oraz
010000000
010000010
....
010000001
010000011
010111111

Przykładowo dla bajtów o prawdopodobieństwach wystąpień 0.35, 0.235, 0.155, 0.15, 0.05, 0.045, 0.01, 0.005 tworzymy binarne drzewo :

0.35	1 1
0.235	1 0
0.155	0 1 1
0.15	0 1 0
0.05	0 0 1 1
0.045	0 0 1 0
0.01	0 0 0 1
0.005	0 0 0 0

Średnia długość kodu w bitach potrzebna do zapisania danych o podanych prawdopodobieństwach napotkania wynosi: 2*0.35+2*0.235+3*0.155+3*0.15+4*0.05+4*0.045+4*0.01+4*0.005=2.52 bita.

Pamiętać należy, iż do zapisania 8 wartości normalnie potrzebujemy 3 bitów, zysk wynosi 16 %..

Porównując te dwie metody, stwierdzono że metoda *Shannon-Fano* daje większe zyski przy dużych różnicach prawdopodobieństw, podczas gdy metoda *Huffmana* staje się bardziej opłacalna przy mniejszych różnicach prawdopodobieństw między elementami podlegającymi kompresji. Często w programach archiwizujących mamy do czynienia z przedefiniowanymi drzewami, tzn. przygotowanymi wcześniej.

Np. częstości występowania liter w różnych językach mogą zostać przeliczone wcześniej i następnie stosowane przy kompresji tekstów. W tym przypadku mamy do czynienia z zyskiem dotyczącym czasu kompresji, ponieważ pozbywamy się problemu tworzenia drzewa dla każdego bloku danych z osobna.

Bardzo znanym algorytmem kompresji jest Imploding. Jest on złożony z dwóch oddzielnych algorytmów. Pierwszy z nich pakuje dane sekwencyjnie używając ślizgającego się offsetu o zakresie 4K lub 8K (dane z PKZIP'a), natomiast drugi jest stosowany do kompresji danych uzyskanych z działania pierwszego algorytmu za pomocą binarnych drzew *Shannon-Fano.*

Imploding używa dwóch lub trzech drzew. Podstawowymi są te, w których koduje się długości sekwencji oraz wskazania offsetowe (dystans), w trzecim drzewie, jeśli jest uaktywnione, koduje się wszystkie dane nie objęte kompresją sekwencyjną. Trzecie drzewo służy do przedstawienia całego zbioru ASCII i zawiera 256 wartości, jeśli jest aktywne, to najmniejsza długość sekwencji (NDS) wynosi 3, w przeciwnym wypadku 2.

Drzewo długości sekwencji zawiera 64 wartości z zakresu od NDS do 63+NDS. Drzewo dystansów zawiera 64 wartości z zakresu od 0 do 63 odpowiadających starszym 6 bitom dystansu. Same drzewa również podlegają kompresji, są spakowane w formacie:

pierwszy bajt oznacza ilość bajtów spakowanego drzewa,
kolejne bajty =>
starsze 4 bity : liczba wartości o podanej liczbie bitów 1..16,
młodsze 4 bity : liczba bitów potrzebna do zapisania wartości 1..16.

Wynikowo blok danych składa się na spakowane drzewa *Shannon-Fano*, a zaraz po nich strumień bitów zawierający kody i dane dla deimplodera.

Algorytm dekompresji Imploding:
początek:
powtarzaj pętlę do końca WY
{czytaj 1 bit z WE
jeśli jest zapalony to
{{jeśli drzewo ASCII jest aktywne to
{odczytaj znak korzystając z tego drzewa}
w przeciwnym wypadku
{czytaj 8 bitów z WE}}
kopiuj bajt do WY}
jeśli zgaszony to
{jeśli offset 8 KB to
{odczytaj 7 bitów dystansu} (*młodszych 7 bitów *)
jeśli offset 4 KB to
{odczytaj 6 bitów dystansu}
używając drzewa dystansów odczytaj 6 starszych bitów dystansu do D
używając drzewa długości odczytaj wartość długości sekwencji do L
L:=L+NDS
jeśli L=63+NDS to
{czytaj 8 bitów z WE i dodaj tą wartość do L }
cofnij się wstecz o D+1 bajtów w WE i kopiuj L bajtów z tej pozycji do WY (*jeśli pozycja ta jest przed początkiem WY to przyjmuje się, że obszar pamięci poprzedzający WY jest wypełniony zerami *)}}
koniec.

Aktualnie istnieje wiele programów wykorzystujących własne, zmodyfikowane algorytmy Imploding, np. w Turbo Implodeтze można ustawiać dowolny offset w zakresie od 128 do 8196 bajtów oraz podawać wielkość bufora służącego do przyspieszenia przeszukiwania pamięci (opisane wyżej). Można powiedzieć, że ile jest programów do kompresji danych, tyle metod, niezmienialne pozostają tylko ogólne zasady, a nie konkretne rozwiązania.

Dotychczasowe metody kompresji, z wyjątkiem metod kodowania względnego, opierały się na pakowaniu bajtów jako najmniejszych jednostek informacji przez nie 'widzianych'. Jednak można by się pokusić o stworzenie cruncher'ów pakujących bity. Takim bitowym kompresorem danych może być opisana poniżej metoda *Filtering.*

Zauważmy, że często w ciągu bajtów zdarza się, iż jakieś bity w kolejnych bajtach nie zmieniają się. Jeżeli teraz oznaczymy wszystkie bity nie zmieniające się (mogą to być zarówno zera i jedynki), tworząc jednobajtową maskę w której jedynka oznaczać będzie bit nie zmienialny, a zero zmienialny, to dodając do tego kilka bitów na licznik ilości bajtów objętych danym rozkazem i dalej umieszczając strumień bitów, które mają być przy dekompresji kolejno wpisywane do bajtów w miejsca określone przez maskę (w miejsca gdzie są zera), możemy uzyskać całkiem ciekawe oszczędności na objętości danych w pamięci.



Załóżmy, że maska ma postać %01011001, a więc zmieniać się będą tylko bity 2,3,6 i 8. Przy dekompresji za bazową wartość bierzemy ostatni bajt, który został przepisany do strumienia wynikowego. Teraz na bazowej wartości wykonujemy logiczne and z maską, przez co wygaszone zostają wszystkie bity przeznaczone do zmiany. Zapamiętujemy wynik w zmiennej np. F.

Pobierając bity ze strumienia wejściowego, dekompresor (w tym przykładzie) dla każdego rekonstruowanego bajta przeznacza kolejne 4 bity i w jakiejś zmiennej bajtowej, np. G, odpowiednio wstawia je na kolejne pozycje 2,3,6 i 8. Wykonując operację logiczną G or F uzyskujemy pierwotną wartość bajtu. Oczywiście musimy zawsze zainicjować licznik, aby wiedzieć ile bajtów jest zakodowanych za pomocą kolejnych rozkazów.

Kiedy przy okazji dodamy kodowanie *Shannon-Fano*, to cały strumień wejściowy może być interpretowany następująco:
początek:
powtarzaj pętlę do końca WE
{pobierz 1 bit jeśli zero to
{pobierz 8 bitów i kopiuj do WY}
jeśli nie zero to
{pobierz odpowiednią ilość bitów do rozkodowania długości sekwencji bajtów jakiej dotyczy rozkaz używając do tego drzewa Shannon-Fano
pobierz 8 bitów maski
wykonaj operację F (opis wyżej)
powtarzaj pętlę do końca długości sekwencji
{wykonaj operację G
kopiuj do strumienia wyjściowego bajt o wartości (G or F) }}}
koniec.

Podsumowując, należy powiedzieć, iż o jakości metody kompresji decydują dwa czynniki. Pierwszym jest oczywiście zysk ze zmniejszenia objętości danych. Drugim jest uniwersalność metody względem różnego typu danych. Uniwersalność ta jest jednak pojęciem względnym.

Jeżeli archiwizujemy programy na dysku to program do kompresji musi 'chwytać' jak najwięcej typów danych, ze względu na ich wielorakość w programach. Natomiast jeżeli typ danych mamy z góry określony, np. grafika, to korzystniejsze staje się stosowanie metod bardziej specjalistycznych, mniej ogólnych, aczkolwiek często bardziej zyskownych.

Jak napisano w pewnej gazetce dyskowej (Mega Magazine), opracowywanie nowych algorytmów do kompresji danych jest chorobą wirusową - nowym wirusem komputerowym, który pożera każdą wolną chwilę programisty. Autor gwarantuje całkowite i nieodwracalne zarażenie spowodowane lekturą tego artykułu.

Artur Wojciechowicz (Tori)

W artykule skorzystano z dokumentacji PKZIP oraz z książki "Data compression" Gilberta Held'a.

Assembler na C64

Tym razem w naszym kąciku początkującego kodera zabierzemy się nareszcie za to, czego pewnie wszyscy już oczekują. Mam na myśli przerwania. Wiemy już jak napisać prostego scrolla. Wiemy również jak zsynchronizować różne operacje na ekranie z jego wyświetlaniem (generowaniem) przez komputer.

Generalnie obowiązuje nas zawsze jedna i ta sama zasada: cokolwiek modyfikujemy na ekranie, musimy to robić w takim momencie, kiedy jesteśmy pewni, że modyfikowany fragment ekranu **NIE** jest właśnie generowany. Jeżeli zapomnimy o tej zasadzie to efekty będą w najlepszym wypadku nieciekawe.

Aby się o tym upewnić, proponuję np. w naszym scroll'u z poprzedniego odcinka zmodyfikować odrobinę wartość z którą jest porównywany rejestr RASTER ($d012) zwiększając ją, dajmy na to, o $04.

Inny przypadek: załóżmy, że umieściliśmy na ekranie sprite'a (chyba wszyscy wiemy co to takiego...). Teraz zaczynamy go przesuwać i jeżeli trafi się tak, że będzie on akurat wyświetlany w tym samym momencie, to na pewno jego kształt na ekranie będzie zniekształcony. Wszystkie te "efekty specjalne" można zaobserwować w wielu starszych grach na C-64.

Szarpiące się scroll'e tudzież sprite'y nie należały kiedyś do rzadkości. My jednak programujemy C-64 w roku 1992. A to do czegoś zobowiązuje. Zabierzmy się zatem do przerwań.

Jak już wcześniej wspominaliśmy, po włączeniu zasilania kontrolę nad komputerem przejmuje jego system operacyjny. Wykonuje on kilka niezbędnych procedur inicjujących rozmaite "podzespoły" - VIC'a, SID'a itp. Następnie sprawdza jeszcze, czy nie został zainstalowany w expansion porcie jakiś samostartujący cartridge.

Odbywa się to w ten sposób, że sprawdzany jest obszar pięciu bajtów od $8004 do $8008 i porównywany z zapisanym w ROM'ie wzorcem. Jeżeli znajduje się tam sekwencja: $C3 $C2 $CD $38 $30, co odpowiada zakodowanemu w standardzie PETASCII tekstowi CBM80, to system pobiera z komórek $8000/$8001 adres pod który przekazuje kontrolę (startuje program cartridge'a).

Oprócz tego, pod adresem $8002/$8003 znajduje się tzw. wektor "warm start" cartridge'a. W przypadku wywołania przerwania NMI (np. wciśnięcia klawisza RESTORE) procesor skacze pod adres wskazany przez ten wektor. Jak nietrudno się domyśleć możemy tą właściwość systemu operacyjnego wykorzystać do własnych celów np. uniemożliwić reset komputera... ale o tym innym razem.

Na razie załóżmy, że nie mamy żadnego cartridge'a więc system po stwierdzeniu tego zainicjuje swoje przerwania i wystartuje zawarty w ROM'ie interpreter języka BASIC. Nas najbardziej z tego wszystkiego interesują przerwania. Istotę przerwań omawialiśmy już w artykule dotyczącym Amigi (KEBAB 7-8/92 str. 20). Wiemy także, że dzięki nim możemy np. korzystać z klawiatury i widzieć na ekranie efekty naszego "stukania w klawisze". Przerwania wykonują dla nas również wiele innych mniej lub bardziej pożytecznych rzeczy (np. "mrugają" kursorem). Jak się to jednak wszystko odbywa?

W C-64 oprócz procesora głównego (CPU), graficznego (VIC) i dźwiękowego (SID), znajdują się jeszcze dwa specjalizowane układy o nazwie CIA. Rejestry tych układów znajdują się odpowiednio w obszarach $DC00-$DC0F i $DD00-$DD0F. CIA to skrót od angielskiego Complex Interface Adapter. Układy te często nazywa się po prostu portami I/O (we/wy).

Nie jest to całkiem słuszne, bo oprócz samych portów, układy te zawierają jeszcze sporo innych pożytecznych rzeczy. A to: zegar czasu rzeczywistego, dwa tzw. timer'y programowalne na rozmaite sposoby oraz rejestry kontroli przerwań. Tak, obydwa układy CIA są w stanie generować przerwania dla procesora.

System operacyjny w trakcie wykonywania w/w procedur inicjujących ustawia jeden z timerów układu CIA1 na odliczanie około 1/60-tej sekundy. Inny rejestr z tego układu jest tak zaprogramowany, że po każdym odliczeniu w/w czasu układ generuje przerwanie IRQ. Nazwa ta to skrót od angielskiego (I)nterrupt (R)e(Q)uest (po polsku: prośba o przerwanie. Sugeruje nam ona, że procesor może, ale wcale nie musi zgodzić się na wykonanie przerwania.

I tak właśnie jest. Na wykonanie przerwania IRQ procesor zgodzi się tylko wtedy, gdy znacznik I z rejestru SR będzie skasowany (patrz też KEBAB 9/92). Dowód na to mieliśmy w naszym scrollu z poprzedniego odcinka cyklu. Dopóki nie ustawiliśmy (komendą SEI) znacznika I, dopóty przerwania systemowe zakłócały nam poprawną pracę programu.

W momencie gdy znacznik ten ma wartość jeden, procesor ignoruje wszelkie prośby typu IRQ. Istnieje jeszcze jeden poziom przerwań dla procesora 6510. Przerwania te nazywają się NMI.

W pełnym brzmieniu (N)on (M)askable (I)nterrupts. Z wykonania przerwań tego poziomu procesor nie może się "wymigać". Natomiast są one stosunkowo mało wykorzystywane przez programistów gdyż istnieją tylko trzy możliwe źródła tego typu przerwań.

Źródła te to układ CIA2, styk "D" z expansion portu oraz... klawisz RESTORE. Z pewnością każdy z nas zna takie sytuacje gdy wydaje się, że komputer "zawisł" a mimo to odpowiednie wciśnięcie (w starszych wersjach: uderzenie) klawisza RESTORE przy jednoczesnie wciśniętym klawiszu RUN/STOP, powoduje przywrócenie kontroli nad komputerem.

Otóż każdorazowe wciśnięcie (uderzenie) klawisza RESTORE powoduje wywołanie przerwania NMI. Niezależnie od stanu znacznika I, procesor musi zaprzestać wykonywania czegokolwiek innego i zająć się przerwaniem. Polega to na tym, że aktualna komenda jest wykonywana do końca a następnie przesyłany jest adres (stan rejestru PC) pod którym aktualnie wykonywany był program, do specjalnego obszaru pamięci zwanego stosem (ang.: stack) w następującej kolejności: najpierw starszy bajt potem młodszy.

Następnie zostaje przesłana na stos aktualna zawartość rejestru SR. Po tych operacjach niezbędnych dla prawidłowej kontynuacji programu po powrocie z przerwania, procesor pobiera zawartość komórek o adresach $FFFA i $FFFB i wstawia je do rejestru PC. Oznacza to skok pod adres zapisany w tychże komórkach. Standardowo jest tam zapisany adres procedury obsługi przerwania NMI w pamięci ROM ($FE43). Procedura ta najpierw przesyła na stos zawartość rejestrów A, X, Y a następnie sprawdza stan klawisza RUN/STOP.

Jeżeli klawisz ten nie był wciśnięty, to następuje natychmiastowy powrót z przerwania jak gdyby nic się nie stało. Jest to widoczne (a raczej niewidoczne) gdy w dowolnym momencie wciśniemy sam klawisz RESTORE. Jeżeli jednak będziemy mieli w tym samym czasie wciśnięty klawisz RUN/STOP to wtedy procesor nie powróci z powrotem do swoich czynności lecz wykona jeszcze kilka procedur inicjujących na nowo m. in. VIC'a. Dlatego np. po wciśnięciu RUN/STOP-RESTORE ekran przybiera swoje zwyczajowe niebieskie kolory.

W przypadku przerwań IRQ, a te przede wszystkim będą nas interesowały w przyszłości, sprawa wygląda podobnie. To znaczy procesor wykonuje do końca komendę, którą był zajęty w momencie pojawienia się impulsu IRQ. Następnie (inaczej niż NMI) sprawdza stan znacznika I. Jeżeli znacznik ten nie był ustawiony, to następuje (tak samo jak NMI) przesłanie na stos aktualnego stanu licznika rozkazów (PC) i rejestru stanów (SR). W tym momencie procesor sam ustawia sobie znacznik I, po to aby w trakcie wykonywania przerwania nie mogło wystąpić kolejne przerwanie. Dzięki takiemu rozwiązaniu wszystkie późniejsze prośby IRQ są ignorowane. Teraz pobierany jest z komórek $FFFE/$FFFF adres procedury obsługi przerwania IRQ i skok do niej ($FF48). Jeżeli weźmiemy jakiś debugger i komendą

```
D FF48
```

dokonamy disasemblacji w/w obszaru pamięci ROM, to powinniśmy zobaczyć coś takiego:

```
FF48 PHA
FF49 TXA
FF4A PHA
FF4B TYA
FF4C PHA
FF4D TSX
FF4E LDA $0104,X
FF51 AND #$10
FF53 BEQ $FF58
FF55 JMP ($0316)
FF58 JMP ($0314)
```

Omówmy zatem wszystko po kolei. Na samym początku mamy komendę PHA. Jest to komenda, która umieszcza na stosie (patrz słowniczek) zawartość akumulatora. Następnie TXA (to już znamy) i ponowne PHA spowoduje, że na stosie znajdzie się dotychczasowa zawartość rejestru X. I podobnie TYA, PHA prześle nam na stos zawartość rejestru Y. Teraz zaczynają się trudniejsze rzeczy. TSX to komenda, która kopiuje zawartość tzw. wskaźnika stosu (ang.: (S)tack (P)ointer) do rejestru X. Jak zwykle skrót od (T)ransfer (S)P to (X).

Jest to potrzebne po to, aby następna komenda (LDA $0104,X) mogła pobrać z obszaru pamięci stosu (proszę zwrócić uwagę, że nie napisałem "ze stosu") umieszczoną tam dopiero co, dotychczasową (sprzed przerwania) wartość rejestru SR. Dokładne wytłumaczenie tego jest dość kłopotliwe i w naszych rozważaniach na temat przerwań, mało istotne toteż ograniczymy się jedynie do podania, że wszystkie cztery komendy od $FF4D do $FF53 służą do sprawdzenia, czy przerwanie zostało wywołane przez hardware'owy impuls IRQ, czy też przez komendę BRK.

W pierwszym przypadku następuje skok (BEQ) pod adres $FF58, w drugim natomiast wykonuje się komenda JMP ($0316). I w jednym i w drugim przypadku mamy do czynienia z nowym (dla nas) trybem adresowania. Proszę zwrócić uwagę na nawiasy otaczające adres komendy JMP. W przypadku adresowania bezwzględnego (bez nawiasów) oznaczałoby to po prostu skok pod adres $0314 lub $0316.

Natomiast w przypadku adresowania tzw. pośredniego (indirect adressing), bo tak się ten tryb nazywa, oznacza to skok pod adres,

na jaki składa się zawartość kolejnych dwóch bajtów od komórki $0314 (lub $0316). Przy czym najpierw odczytywany jest młodszy a następnie starszy bajt adresu. Trochę to zawiłe więc najlepiej pokażmy to na przykładzie. Obejrzyjmy naszym debugger'em pamięć od adresu $0314.

```
M 0314
```

powinno nam dać w odpowiedzi następującą linijkę:

```
:0314 31 EA 66 FE 47 FE 4A F3
```

Jak widzimy, pod adresem $0314 mamy wartość $31. Pod adresem $0315 mamy $EA. Te dwa bajty to zakodowany w standardowym dla procesora 6502 systemie lo/hi byte (młodszy/starszy bajt) adres $EA31. Proste? Najpierw młodszy bajt ($31), następnie starszy ($EA). Jak złożymy je razem to mamy wspomniane już $EA31.

Wracając jednak do naszego JMP ($0314) to pewnie już się domyślamy, że pod adresem wskazanym przez te dwie komórki musi znajdować się w ROM'ie procedura obsługi przerwań. Po co tyle zagmatwania? Ktoś mógłby zapytać. Jedno przerwanie i aż tyle zbędnych skoków po pamięci. Otóż nie jest to zupełnie takie bezsensowne.

To, że przy obsłudze przerwania procesor najpierw pobiera adres skoku spod $FFFE zostało zaprojektowane przez twórców samego 6502/10. Działanie procesora w tym momencie można porównać do skoku pośredniego:

```
JMP ($FFFE)
```

Twórcy C-64 wstawili w ROM'ie wartości umożliwiające skok pod adres $FF48. Dlaczego? Ponieważ sam procesor, gdy otrzyma impuls IRQ, odkłada na stos tylko aktualny stan licznika rozkazów (PC) oraz stan rejestru SR. Aby jednak przerwany program mógł później funkcjonować poprawnie tj. tak, jak gdyby nic się nie wydarzyło, należałoby jeszcze uchronić gdzieś zawartość rejestrów X, Y, A. To właśnie robi procedura rozpoczynająca się od $FF48.

Po wykonaniu wszelkich czynności związanych z rejestrami następuje skok pośredni JMP ($0314) do procedury obsługi przerwań pod adres $EA31. Czemu tak? Nie prościej skoczyć bezpośrednio używając komendy JMP $EA31? Owszem! Byłoby prościej. Ale nie o to w tym wszystkim chodzi. Dzięki temu, że został wykorzystany rozkaz skoku pośredniego, możemy sobie zaprogramować własną procedurę, która następnie może być wykonywana każdorazowo w miejsce (lub oprócz) standardowej procedury z pamięci ROM.

Jak to zrobić? To bardzo proste! Przecież komórki $0314 i $0315 znajdują się w pamięci RAM! Możemy w takim razie wpisać tam co nam się tylko podoba... No! Może nie tak całkowicie co nam się podoba, ale... Załóżmy, że napisaliśmy jakąś procedurę (podprogram) i chcemy aby wykonywała się ona co każde przerwanie (najczęściej ok. 50 razy na sekundę). Procedura nasza leży w pamięci począwszy od adresu $6000. Napiszmy zatem:

```
A5000 SEI
A5001 LDA #$00
A5003 STA $0314
A5006 LDA #$60
A5008 STA $0315
A500B CLI
A500C RTS


A6000 INC $D020
A6003 JMP $EA31
```

Teraz jeszcze SYS 5*4096 i już jest! Nasz pierwszy program wykorzystujący przerwania działa! Co prawda niczego poważnego jeszcze nie robi, ale jest to chyba najprostszy sposób aby uświadomić wszystkim naszym kursantom "jak to działa". Przyjrzyjmy się jeszcze trochę dokładniej kolejnym komendom.

Już na samym początku mamy SEI. Co ta komenda oznacza to już wiemy. Ale po co ona jest tu potrzebna? O tym za chwilę. Teraz następne dwie komendy. LDA #$00 i STA $0314. Już wiemy, że pod adresem $0314 mieści się młodszy bajt wektora procedury przerwań. Tam wstawiamy młodszy bajt adresu naszej procedurki.

A teraz uwaga! wyobraźmy sobie taką sytuację, że procesor właśnie wykonał te dwie komendy i w tym momencie otrzymał impuls IRQ. Co się dzieje? Procesor wykonałby stadardowo skok poprzez wektor $0314/15. Ale młodszy bajt tego wektora został już przez nas zmieniony. W rezultacie mamy pod $0315 wartość $EA, a pod $0314 wartość $00. W efekcie takiego działania procesor wykonałby skok pod adres $EA00 czyli w przysłowiowe maliny. A dokładniej w środek procedury czyszczenia linii ekranu.

Nam jednak wcale nie o to chodziło. Aby takiej sytuacji uniknąć, wstawiamy na początku naszego programu komendę SEI. W takim przypadku, dopóki nie ustawimy całego wektora jak należy, procesor będzie ignorował wszystkie prośby o przerwanie. Dobrze! Następne dwie komendy to wstawienie starszego bajtu adresu naszej procedury. Na końcu jeszcze tylko CLI (bo już wszystko mamy ustawione) i przerwania już chodzą, a nasz program inicjujący powraca do BASIC'a komendą RTS.

Pozostaje nam jeszcze tylko przyjrzeć się co robi fragment programu leżący pod $6000. Właściwie nic mądrego. Jeden INC $D020 powoduje "mruganie" borderem a JMP $EA31 to skok do oryginalnej obsługi przerwań. Dzięki temu ostatniemu nasz komputer nadal działa "normalnie" tzn. oprócz zmieniających się kolorów ramki dlaej działa klawiatura, kursor etc.

Oczywiście pozostaje tu pole do eksperymentów dla wszystkich Czytelników. Zamiast INC $D020, możemy wstawić znacznie dłuższy program i zakończyć go skokiem pod $EA31.

W następnym odcinku omówimy już przerwania rastrowe i zabierzemy się ponownie do poważniejszego kodowania. Zapraszam!

SD!

Słowniczek:

Sprites - to niezależne obiekty graficzne, które można umieszczać na ekranie C-64. Pozycja tych obiektów jest definiowana w odpowiednich rejestrach VIC'a. Inne rejestry odpowiadają za to czy sprite ma się pojawiać "przed" czy "za" pozostałą zawartością ekranu.

Timer - specjalny rodzaj zegara. Tzw. zegar odliczający w dół. Zegar taki po ustawieniu odpowiedniego czasu odlicza do zera.

Wektor - tu: dwie komórki pamięci, przeznaczone do przechowywania adresu jakiejś procedury systemowej. W C-64 istnieje wiele wektorów, gdyż dla większości procedur systemowych przewidziano odpowiedni wektor. Dzięki takiemu rozwiązaniu programista może opracowywać własne procedury i, korzystając z wektorów, integrować je z systemem operacyjnym. Oprócz wektorów przerwań IRQ, BRK, NMI, istnieją także wektory dla procedur LOAD, SAVE, OPEN, CLOSE, CHROUT, STOP i inn.

Stack - po polsku: stos. Specjalny fragment pamięci w obszarze adresowym procesora. Specjalne komendy (PHA,PHP) umożliwiają szybkie odkładanie na stos zawartości rejestrów. Inne natomiast (PLA,PLP) służą do szybkiego pobierania odłożonych tam wartości. Nazwa pochodzi od sposobu dostępu. Zasada działania jest podobna do np. stosu kartek. Ostatnią kartkę położoną na stos, możemy pobrać jako pierwszą. Jeżeli odłożyliśmy pięć kartek w określonej kolejności, to pobrać je musimy w kolejności odwrotnej. Oczywiście zawsze instnieje możliwość odczytania jakiejś kartki spomiędzy innych, ale to już nie należy do tematu. Dla naszych rozważań przyjmiemy, że kartki (bajty) możemy odkładać tylko na szczyt stosu i stamtąd je pobierać. W C-64 znajduje się zarezerwowany na ten cel obszar od $0100 do $01FF. Specjalny rejestr SP (Stack Pointer) czyli wskaźnik stosu służy procesorowi do szybkiej orientacji w całym obszarze stosu.

Sprawozdanie z pola walki...

Największym wydarzeniem w powakacyjnej Polsce było warszawskie Copy Party, zorganizowane przez grupę Action Direct. Copy Party to nic innego jak wielki zlot użytkowników Amigi, począwszy od skromnych lamerków, na elitarnych grupach kończąc. My, jako redakcja Kebaba, byliśmy tam również i niniejszym spróbujemy wszystko opisać. Zanim przejdziemy do szczegółów, należy powiedzieć, iż mimo kilku niedociągnięć organizacyjnych, a także paru niemiłych wydarzeń (ze strony przyjezdnych), impreza udała się, ludzie byli całkiem zadowoleni, nie było ani pożaru, ani powodzi.

Party rozpoczęło się w piątek o 17, lecz my przybyliśmy na miejsce dopiero w sobotę rano. Większość Amigowców spała, jedynie na warcie stał, a raczej siedział Docent. Już na początku pierwsza duża krecha dla organizatorów: okazało się, iż mimo wcześniejszej rezerwacji, sali dla nas nie ma. W końcu dolinkowaliśmy się do sali zajmowanej przez szczecińską grupę **INF**. Party odbywało się w szkole, więc miejsca było dość. Była dostępna nawet sala gimnastyczna dla lubiących koszykówkę.

Na parterze umiejscowiono big screen'a i całą aparaturę nagłaśniającą, natomiast pierwsze piętro było przeznaczone na "rozbicie się" ze sprzętem. Do czasu rozpoczęcia się pierwszego music competition (konkursu na najlepszą muzykę), czas upływał na nawiązywaniu znajomości, oraz kopiowaniu programów.

Pojawiło się wiele nowych dem zagranicznych, ciekawych użytków, nowych obiektów do np. Imagine'a. Grami mało kto się zajmował, chyba że w celach rekreacyjnych, po długiej pracy. Na party obowiązywała zasada: "Co ty masz, to i ja zaraz będę miał", nie było więc mowy o sprzedawaniu programów, a całość odbywała się w ramach koleżeńskiej współ pracy. Nie wszyscy jednak tak miło spędzali czas.

W sztabach grup **Action Direct, Deform i W.F.M.H.** trwały prace nad składaniem nowych dem. O ile Musashi z Deformów, oraz Robin i Mr. Soft z W.F.M.H. po wielogodzinnej harówce dema swe złożyli, to zabiegi Action Direct skończyły się tylko na rozpaczy ich muzyka - XTD.

Na otarcie łez, A. Direct wydali długo oczekiwango diskmaga, a mianowicie Zig Zaga #6. Na parterze Mr.Root z **Sanity** umilał czas grą na syntezatorze. Mimo naprawdę wspaniałej muzyki, dzięki maksymalnie nastawionej głośności, mało było śmiałków słuchających "koncertu" dłużej niż przez kilka minut. Tak mniej więcej upływał czas do godz. 16, czyli do rozpoczęcia się music compo.

Do konkursu, po obradach specjalnego jury, dopuszczono 14 muzyczek. Puszczano je kolejno, przy czym cały czas na big screenie pokazywany był odgrywający je ProTracker. Każdy otrzymał kartę do głosowaniana, na której miał wypisać kolejno pięć jego zdaniem

najlepszych muzyczek. Umieszczenie na jednej karcie do głosownia trzech pozycji (muzyka, grafika, demo) nie było zbyt dobrym pomysłem. Po wpisaniu numerów modułów, trzeba było czekać do godziny 24, aby już do końca wypełnioną kartę móc oddać. Bardzo nam się podobało podawanie informacji o czasie trwania modułu i zatajenie nazwiska/ksywy jego wykonawcy. Głosowano więc na muzykę, a nie na wykonawcę. Music Compo nie cieszyło się zbyt dużą frekwencją, wielu narzekało na poziom, ale według mnie nie było to winą piszących moduły (przeważnie), gdyż inaczej słucha się muzyki gdy jest ona puszczana w rozsądnej głośności, a inaczej gdy wręcz zgniata ona głowę. Competition i 1 mln złotych wygrał **Jakub Husak.**

Korzystając z czasu do rozpoczęcia się graphic compo, udaliśmy się do największej w Warszawie oazy lamerstwa, czyli na giełdę przy ulicy Grzybowskiej. Tłumy biegające w pogoni za nowymi grami, przemądrzający się "koderzy" (fani RSI demomaker`a), to tam normalne zjawisko. Niedobrze się robi również na widok sT-ciaków. Szybko więc wróciliśmy na nasze kochane party.

Tymczasem na big screenie organizatorzy prezentowali film "Space Wars". Jest to parodia filmu pt."Gwiezdne wojny", wykonana w 100% za pomocą Amig. Wspaniałą animację stworzył Tobias Richter, a muzykę skomponował Bjorn A. Lynne. Używali oni pięciu Amig w tym A-2000 z procesorem 68030/50 MHz i A-3000/25 MHz.

Film w humorystyczny sposób przedstawiał atak rebeliantów na Gwiazdę Śmierci, gdzie roztrzygającą bronią była puszka Coca-Coli. Jakość obrazu dzięki karcie DCTV była na tyle dobra, by pościnać z nóg wszystkich Amigowców, nie mówiąc już o zwykłych zjadaczach chleba.

Ciekawostką jest tu fakt, iż stworzenie takiego filmu nie wymagało użycia magnetowidów z możliwością nagrywania pojedynczych klatek (!), gdyż cała animacja była zgrywana bezpośrednio na taśmę.

O dwudziestej rozpoczęło się graphic competition, na którym pokazano ponad 30 rysunków kilkunastu grafików. Każdy mógł zaprezentować maksymalnie dwa obrazki i podobnie jak w music compo nie można było podpisywać się pod grafiką.

Zachęta Mr.Roota, aby piętnować gwizdami rysunki wykonane w technice raytracingu lub skanowane, wywołało zabawny efekt: widzowie krzyczeli "skaning !!!" nawet gdy na ekranie widoczne było jedynie okno AmigaDos'u.

Poziom był wysoki, ale naprawdę w ciągu kilku chwil trudno jest odróżnić precyzyjne, ręczne rysowanie, od zamaskowanego skaningu (a podobno miało być fachowe jury eliminujące tego typu prace ?!). Bardzo podobał mi się prosty, ale niezwykle efektowny rysunek przedstawiający Don Kichota: zaledwie kilka precyzyjnych pociągnięć, a efekt imponujący. Graphic Compo i 1 mln złotych wygrał **Animal** z grupy **Action Direct.**



Sporo emocji dostarczył pojedynek pomiędzy uczestnikami Copy Party, a nieliczną grupą iBM'owców. Wytoczono ciężką artylerię: IBM PC 486/50 MHz z 24-bitową kartą graficzną, kartą muzyczną, CD-Rom'em (czyli trudno by było o lepszą konfigurację) naprzeciw Amigi 4000 bez żadnego dodatkowego osprzętu. Mimo takiej dysproporcji Amiga dzielnie obroniła się: ma przecież standardowo 16,8 milionową paletę (ponad 256 tysięcy kolorów jednocześnie na ekranie), ponadto choćby 4 przetworniki analogowo-cyfrowe są wyżej "notowane" nawet od szesnastu zwykłych generatorów dźwięku, a o systemie operacyjnym nie warto nawet wspominać.

Ciekawe czy następnym razem ktoś wystawi Amisię uzbrojoną już w np. Opal Vision, kartę z szesnastoma szesnastobitowymi (ale to brzmi...) przetwornikami do dźwięku? Niestety tym razem nasz obóz nie mógł pochwalić się posiadaniem dema napisanego specjalnie dla "czterotysiączki", natomiast pececiarze byli lepiej w nie zaopatrzeni.

Pod względem oprogramowania użytkowego, Amigowców zadziwiła wreszcie normalna (w miarę szybka) praca pod systemem Windows. Nasuwa się pewne pytanie: czy, trzeba było aż 486/50 MHz, aby okienka pracowały jak na zwykłej Amidze 500??? Każdy kto widział Windowsy na AT lub nawet na 386, wie co to znaczy tragedia.

"Kloniarze" nie popisali się również wiedzą, twierdząc, że ich cudeńko jest w stanie wyświetlić na raz całą 24-bitową paletę. Panowie!!! Nawet gdyby każdy piksel (taka kropeczka na monitorze) był w innym kolorze, to przy rozdzielczości 1024x1024 ujrzymy niewiele ponad milion kolorów.

Kulminacyjnym momentem Copy Party był oczywiście "gwóźdź programu" -

Demo Compo. Jeszcze przed godziną planowego rozpoczęcia, wszystkie miejsca były pozajmowane, a ludzie w tylnych rzędach musieli stać, by cokolwiek zobaczyć. Do konkursu wystawiły swe produkcje praktycznie wszystkie większe grupy. Poziom dem był bardzo wysoki, a publiczność stanęła na wysokości zadania, nagradzając najlepsze efekty brawami. Nie obyło się również bez śmiechu nad niektórymi archaicznymi częściami, ale to wszystko podnosiło jeszcze napięcie. Ściszono wreszcie muzykę do głośności zalecanej przez Polskie Towarzystwo Słuchologiczne. Głosowano na 3 dema: pierwsze otrzymywało 3 punkty, drugie dwa , trzecie jeden.

Wygrało przewagą kilkudziesięciu punktów demo "Software" grupy **World Federation of Mad Hackers (W.F.M.H.)**, przed **Old Bullsami** i grupą **Suspect**. Na kolejnych pozycjach uplasowały się:

4. Flying Cows Inc. 5. Deform 6. Mad Elks 7. Investation 8. Defensywa 9. Beermacht 10. Barbie Fuckers

Niemiłą niespodzianką był fakt, iż członkowie grupy **Old Bulls** po raz kolejny, pomimo wielokrotnych zakazów, głosowali na siebie. Choć nie zmieniło to kolejności, za karę unieważniono ich głosy. Główną nagrodą były 2 mln złotych, lecz całość procedury ogłaszania wyników i wręczania nagród budziła jak najbardziej krytyczne odczucia. Chociaż wyniki wszystkich trzech konkursów były znane wkrótce po zakończeniu demo compo, oficjalnie podano kolejność dopiero rano.

Zabrakło jakiegokolwiek oficjalnego i choćby troszeczkę uroczystego ogłoszenia wyników i wręczenia nagród. Ponadto oprócz gratyfikacji pieniężnych, organizatorzy powinni pomyśleć o jakiś skromnych pamiątkach. Na wcześniejszej takiej imprezie w Żywcu, zwycięzcy otrzymali statuetki wykonane ze zwykłej śruby (!), tanie, ale jakże efektowne...

Na tym właściwie powinien zakończyć się opis Copy Party, gdyż niedziela była już tylko dniem odjazdów. Warto może jeszcze wspomnieć o kilku dodatkowych "efektach ubocznych":

a) Grupie Defensywa skradziono Amigę z 2.5 MB RAM. Mamy nadzieję, że sprawca zostanie ujęty i osadzony w jednej celi z aTari ST lub pecetem. Ponadto zginęło kilka stacji dysków i wiadro "płaskokrążków informacjonośnych".



b) Po imprezie zostało tyle śmieci, że tylko wypada nam wyrazić głębokie wyrazy współczucia organizatorom.

c) Gratulujemy szczęśliwemu znalazcy Action-Replay'a w pisuarze. My obeszliśmy wszystkie szalety, ale sukcesu nie udało się powtórzyć.

d) Na piętrach walały się profesjonalne resztki egzemplarzy pewnej gazety poświęconej Amidze.

e) Odbył się konkurs na najładniejsze beknięcie, ale wyniki nie są nam znane, bo gdy wyjeżdżaliśmy ostatnie beknięcie jeszcze trwało.

f) Było fajnie, ale być może w przyszłym roku będzie jeszcze lepiej...

Wysłannik



LISTY DO I OD REDAKCJI

Droga Redakcjo!

Posiadam Amigę 500. Jestem zupełnie początkującym programistą w asemblerze. Wpisłem listing 2 zamieszczony w KEBABie nr 2-3 na asemblera MASTERSEKA V1.71. Po uruchomieniu pojawia się czarny obraz i nic więcej. Chciałbym, abyście mi odpowiedzieli, dlaczego nie pojawia się sprite sterowany myszką?

Z poszanowaniem G.M.

Co możemy odpowiedzieć? W zasadzie jedynym wytłumaczeniem jest to, że listing musiał zostać błędnie przepisany bądź nieodpowiednio zasemblowany. Sam listing jest poprawny (zostało to sprawdzone). Przy przepisywaniu listingów w jakimkolwiek języku należy niezmiernie uważać aby nie popełnić najmniejszego błędu.

Jakakolwiek nawet "mało istotna" różnica pomiędzy tym co jest wydrukowane a tym co przepisaliśmy spowoduje w najlepszym przypadku błędne działanie programu. W przypadku języka asemblera prowadzi to najczęściej do odwiedzin GURU na naszym ekranie. Inną ewentualnością może być wspomniany wyżej "czarny ekran". Nasza rada to sprawdzić DOKŁADNIE linijka po linijce każdy wiersz programu. Ewentualnie parametry asemblera MasterSeka (znaczenie komend można znaleźć w KEBAB'ie nr 6/92).

Ponadto musisz upewnić się, że zasemblowany kod procedury znajduje się w pamięci CHIP; (opcja "c" przy uruchamianiu assemblera)

Z zadowoleniem przyjąłem listing programu "Tape-Burger"... . Po pracowitym i dokładnym "wklepaniu" za pomocą monitora i uruchomieniu programu, niestety nie chciał on działać. Ponieważ jestem przekonany o poprawnym poprawnym wprowadzeniu zbioru do pamięci proszę o ponowne przejrzenie go przez autora i ewentualną korektę. Moja druga prośba do redakcji. Niektóre systemy "turbo" po odczytaniu nagłówka zbioru podają adres startowy i końcowy zbioru. W jaki sposób go odczytać, jeżeli "turbo" nie ma takiej możliwości lub program nie jest nagrany w systemie turbo?

Z poważaniem

K. Wawrzyniak

Szanowny Panie,

zgodnie z Pańską prośbą autor ponownie przejrzał listing programu i upewnił się co do bezbłędnego wydruku w KEBAB'ie. Dodatkowym na to potwierdzeniem są liczne telefony od czytelników zadowolonych z poprawnego działania programu. Jeżeli jest Pan całkowicie przekonany o poprawnym jego wprowadzeniu, to jedynym wytłumaczeniem może być podłączony do komputera niekompatybilny cartridge np. krajowe cartridge typu "X" lub "Black Box" mogą powodować błędy w działaniu wielu programów m.in. naszego "Tape-Burger'a".



Sprawa druga: większość systemów przyspieszających "turbo" korzysta ze standardowego obszaru pamięci znanego jak tzw. bufor magnetofonu (TBUFFER) rozpoczynającego się od adresu $033C i sięgającego do adresu $03FB.

Zarówno w przypadku korzystania ze standardowego (okropność!) systemu ładowania z taśmy jak i w przypadku wszystkich programów "turbo" wzorujących się na pierwszym

c. d. na stronie 17

Crunchery - który lepszy ?

Dysk na który możemy nagrywać programy, ma jak powszechnie wiadomo 880 kB, jednak po sformatowaniu go uzyskamy już "tylko" 855952 bajty wolnego miejsca, dostępnego bezpośrednio dla nas. Często pracując z dyskami, przytrafi nam się sytuacja, kiedy zabraknie na nich wolnej przestrzeni i mimo uporczywych starań, prób negocjacji, gróźb i przekleństw, Amiga nieubłaganie będzie wyświetlać komunikat "Volume "NaszDysk" is full" (dysk pełny). Mamy wtedy w zasadzie dwa wyjścia:

1. Skasować niepotrzebne pliki (ale co wtedy, gdy wszystkie okażą się potrzebne?).

2. Użyć jednego z programów do kompresji plików.

Pierwszy sposób nie wymaga żadnych komentarzy, warto natomiast zająć się drugim punktem. Zagadnienie dotyczące walki o miejsce na dysku, nie należy do dziedzin, w których wymagana jest wielka wiedza, lecz jeżeli się już tego podejmujemy, to należy mieć ogólne pojęcie co wolno crunch'ować, a czego nie, jakich programów najlepiej użyć i jakie warunki należy spełnić, aby spakowany program dał się uruchomić.

W tym artykule zajmiemy się przede wszystkim zastosowaniami praktycznymi tego cudownego wynalazku jakim jest kompresja.

Podstawowe zagadnienie to wybór odpowiedniego programu. Program ten musi korzystnie zaprezentować się na polu szybkości, łatwości obsługi, ilości dostępnych opcji, a co najważniejsze: wydajności. Oceniając dostępne programy pod tym kątem, na placu boju pozostają trzy: Turbo Imploder, Power Packer i Crunch Mania. Najwięcej cruncher'ów padło przy ocenie ich szybkości - czas kompresji pliku o długości 100 Kb mógł trwać nawet kilka godzin!!! Ze zwycięskich programów, każdy ma swe wady i zalety, spróbójmy więc je krótko omówić.

Turbo Imploder - szybki, efektownie wykonany i bardzo wydajny. Jeżeli zastosujemy program o nazwie *earlyexplode* np. uruchamiając go w startup-sequence, to możemy implodować wszystkie biblioteki, fonty, itp. Po zakończeniu pakowania pliku, zostajemy o tym 'dźwiękowo' zawiadomieni. Dekompresja jest bardzo szybka i chwilkę po wczytaniu programów ze stacji, są one gotowe do pracy. Aby była ona możliwa, w katalogu LIBS musi znajdować się biblioteka "explode.library".

Jest to sporą zaletą Implodera, ze względu na to, iż w przypadku crunch'owania wielu plików na dysku, będą one dodatkowo krótsze o ok. 1600 bajtów każdy. Wystarczy uruchomić jeden zimplodowany plik, aby biblioteka "explode.library" się uaktywniła i dokonywała następnych dekompresji. Turbo Imploder ma również i wady. Zanudza nas beznadziejną muzyczką (moja praca z Imploderem zaczyna się od jej wyłączenia), za każdym razem odczytuje katalog z dysku (zamiast go zapamiętać), nie kompresuje plików z danych (data), zajmuje sporo miejsca na dysku - ok. 64 Kb po skompresowaniu.

Power Packer - również szybki, efektownie wykonany, choć odrobinę mniej wydajny. Przy zastosowniau *PPLoadSeg*, można pakować biblioteki, fonty, itp. Dekompresja wgranego pliku jest już trochę dłuższa, za to możemy pakować moduły, grafikę, animację, oraz teksty! Pomocne stają się wówczas programy takie jak *PPPlay* do odtwarzania modułów, *PPmore* do czytania tekstów, *PPshow* do grafiki i *PPAnim* do animacji.

Ponadto jest wiele, a powstaje coraz więcej, programów dających sobie radę ze spakowanymi danymi, np. *Protracker* czy *Disk Master 3.3*. Jeśli chodzi o kompresję danych, Power Packer stał się już po prostu standardem. Nie należy zapomnieć o bibliotece "powerpacker.library" w katalogu LIBS.

Ogromną zaletą Packera jest możliwość dekompresji plików scrunch'owanych uprzednio przez inne programy, możliwość ikonifizacji, ustawienia własnych preferencji, korzystania ze script'ów i portu Arexx'a. Program nie posiada poważniejszych wad, z wyjątkiem jednej - jest komercyjny!

Crunch Mania - szybki, krótki (ok.13 Kb po spakowaniu), z szybkim decrunch'erem, oraz cechujący się niezwykłą wydajnością. Mamy możliwość ikonifizacji, ponadto program podaje nam stale ilość wolnej pamięci Jego wadami są: podawanie długości plików i efektywności kompresji w postaci heksadecymalnej i miganie diody w podczas jego pracy (przeszkadza to w słuchaniu w tym czasie muzyki), brak opcji kompresji danych, oraz niemożność wgrania pliku, dekompresji i nagrania go z powrotem na dysk w normalnej (rozpakowanej) postaci.

W najnowszej wersji Crunch Manii - 1.6 (która zresztą dotarła do redakcji już w czasie pisania niniejszego artykułu), zostały poprawione wszystkie uchybienia pop-

rzednika. Ponadto zastosowany został nowy, niespotykany dotąd sposób obsługi spakowanych danych. Dzięki programowi *RTDD* możemy kompresować nie tylko teksty, moduły, itp., ale i nawet ikony(!).

Wydajność powyższych programów przedstawia tabelka:

PROGRAM	NORMALNIE	CRUNCH MANIA	TURBO IMPLODER	POWER PACKER
Asm One	88200	50560	50636	51360
BootX	64048	33912	34604	34836
Cygnus	110228	58776	59612	60752
DiskMaster	56020	29680	29872	30560

Efekt kompresji jest zależny od posiadanej wersji programu. Tabela została wykonana na bazie następujących wersji: Crunch Mania - v1.6, Turbo Imploder - v4.0, Power Packer - v4.0.

Po tym podsumowaniu, należy rozwiązać tytułowy problem - którego z nich najlepieiej jest używać. Odpowiedź na to pytanie jest złożona. Do kompresji danych możemy korzystać jedynie z Power Packera, lub nowej Crunch Manii. Faktem jest, iż Power Packer był gorszy w wydajności przy programach wykonywalnych, lecz udanie rewanżuje się na polu pracy z plikami data, co w zasadzie rozwiązuje wszelkie wątpliwości.

Do plików wykonywalnych osobiście proponuję Turbo Implodera, ze względu na to, iż program ten stał się już podobnie jak Power Packer standardem, jego pliki są rozpoznawane przez m.in. BootX'a, a wydajność nieznacznie tylko ustępuje Crunch Manii. Skompresowane biblioteki, fonty,itp. po zastosowaniu earlyexplode'a działają idealnie, czego nie można czasami powiedzieć o PPLoadSeg, czy RTDD. Natomiast po właśnie Crunch Manię, warto sięgnąć w sytuacjch krytycznych (gdy na przykład zależy nam na kilku kilobajtach na dysku).

Na zakończenie kilka uwag :

1. Jeżeli posiadasz mało pamięci (najczęściej jest to 0.5 Mb, choć czasami dla niektórych 4 Mb to mało), to pamiętaj, że aby możliwa była dekompresja, wymagana jest wolna pamięć często dwukrotnie większa od rozmiaru spakowanego pliku. I tak na przykład zimplodowany Imagine na Amidze z 1 Mb, nie ma najmniejszego zamiaru się uruchamiać. W tego typu przypadkach należy postarać się o niespakowane wersje (orginalne programy nie są crunchowane),lub też dokupić pamięć.

Innym wyjściem z sytuacji jest posłużenie się *Titanics Cruncher'em.* jego działanie jest inne niż pozostałych programów. Otóż nie kompresuje on pliku jako całości, lecz dzieli go na części i każdą z osobna pakuje. Przy ładowaniu pliku, części te (o długości ok. 5 Kb) są odczytywane i kolejno dekompresowane, przy czym ze względu na ich długość nie jest wymagane wiele pamięci.

2. Jeżeli skompresujemy plik zarażony wirusem linkowym (doczepiającym swój kod do innych programów), to programy antywirusowe, mimo najszczerszych chęci, nie będą potrafiły zlokalizować mikroba, mimo że oczywiście rozpoznają go w pamięci. BootX potrafi zdekompresować plik z Power Packera, ale sprawa jest beznadziejna, gdy użyto np. DefJam Packer'a, czy chociażby naszej Crunch Manii.

3. Czasami zdarzyć się może, że program po sesji z Power Packerem nie będzie działał prawidłowo (herezje z allokacją pamięci), lub nawet wcale. Jest to bardzo, bardzo rzadkie, ale jeżeli już się zdarzy, używamy po prostu innego cruncher'a.

4. Najlepiej kompresują się teksty, pliki wykonywalne zależnie od ich struktury, za to moduły kurczą się bardzo opornie.

Zbigniew Piotrowicz





LISTY DO I OD REDAKCJI

dokończenie ze str. 15

programie Stephana Senz'a, po wczytaniu nagłówka (ang.: header) znajdują się w tym obszarze, oprócz nazwy zbioru, również informacje o jego typie jak i adresach początkowym i końcowym. Czy zostaną one wyświetlone na ekranie w odpowiednim momencie czy nie, zależy tylko od intencji autora danego "turbo". Jeżeli chcemy koniecznie wiedzieć pod jaki adres ma się dany zbiór ładować, wystarczy po wczytaniu nagłówka odczytać pięć pierwszych bajtów z bufora magnetofonowego.

W standardowym systemie pierwszy z nich oznacza typ zbioru, dwa następne to adres początkowy zapisany w ogólnie przyjętym w C-64 formacie młodszy/starszy bajt. Dwa następne to tak samo zapisany adres końcowy zbioru. Jeżeli posługujemy się wprawnie heksem (szesnastkowym sposobem liczenia), to jedno spojrzenie monitorem pamięci pod adres $033C powinno powiedzieć nam wszystko.

Z poziomu BASIC'a należy przeliczyć sobie adresy na dziesiętne korzystając np. z poniższego przykładu.

```
PRINT PEEK(829)+256*PEEK(830)
```

wyświetli nam adres początkowy a:

```
PRINT PEEK(831)+256*PEEK(832)
```

adres końcowy.

W wielu systemach "turbo" typ zbioru znajduje się za adresami ładowania więc należy odpowiednio dopasować wyżej podane linie.

Szanowna Redakcjo !

Od niedawna jestem użytkownikiem Amigi. Używam Anosa 1.3 do pisania programów. Nurtuje mnie następują-

c. d. na stronie 18

Katharsis

Warszawska grupa Katharsis została założona w roku 1990 przez **Raf'a i War'a**. Rozpoczęli oni swą pracę na Amidze od wydania intra, co zresztą z biegiem czasu stało się ich specjalnością - obecnie po Polsce krążą cztery różne wersje intr. Po jakimś czasie szeregi zasilili **Andy Brent** i **S.S Captain**. Następną produkcją było demo o nazwie **Unibobs**, plasujące się na piątym miejscu dyskowego Kebaba. Niestety War nie napisał już następnego dema. Zginął w tragicznym wypadku, co z pewnością zubożyło polską scenę.

Największy rozgłos grupie Katharsis przyniosła działalność S.S Captain'a. S.S Captain był pierwszym w Polsce modem traderem pracującym w trybie 14400 bodów. Zdecydowana większość nowości pochodziła właśnie od niego, a wszystkie przybytki typu giełd, czy miejskich nagrywalni, w dużej mierze na nich bazowały. W ten sposób grupa zyskała sławę wśród nie tylko ludzi zainteresowanych polską sceną, ale i przeciętnych łamaczy joysticków.

Kolejnymi członkami grupy byli koder **Mac** i grafik **Jerry**. Od razu zabrali się oni do pracy, co wkrótce zaowocowało wydaniem nowego dema o nazwie *Let's Strip*. Ówczesna działalność Katharsisów to nie tylko dema i "świeże downlod'y", ale ze względu na wszechstronność ich prezesa - Rafa, również dyski muzyczne - *The Best of Raf.* Ponadto Mac zajął się udoskonalaniem Virus Experta oraz Disk Mastera kodowanego wcześniej przez Gully'ego.

Praca nad Virus Expertem zakończyła się wydaniem komercyjnej wersji polskiej tegoż programu. Swego czasu był to najlepszy "użytek" antywirusowy, lecz ze względu na brak kolejnych ulepszeń, został wyparty przez BootX'a. Wkrótce Warszawiacy stworzyli swą śląską sekcję, do której weszli ex. członkowie Hypnosis. Sekcja ta wydała pod szyldem Katharsis demo *Testament.* Radość z prężnego oddziału śląskiego nie trwała długo. Połowa sekcji zrezygnowała ze współpracy, a został jedynie **Mad Max** i **Don Kichot.**

Dosyć głośną sprawą była wojna pomiędzy Katharsis, a Luzersami. Jeżeli ktoś jest zainteresowany, odsyłamy do Zig-Zaga. Ze swej strony uważamy, iż tego typu konflikty są niepoważne (patrz artykuł Savoir-vivre z poprzedniego numeru).

Do kolejnych ich zadań zaliczyć należy łamanie zabezpieczeń i wydawanie wraz z grupą Action Direct Zig-Zaga. Kod do niego napisał Raf (numery #2-#5). Katharsisi odwiedzali trzykrotnie imprezy typu Copy Party: dwukrotnie zorganizowane na Węgrzech (na jednym z nich pierwsze miejsce zajął będący przez pewien czas ich członkiem - **Mr.Root**) i raz Amega Party w Norwegii.

Grupa pięła się coraz wyżej, lecz nastąpił kryzys spowodowany nieporozumieniami przy przyjmowaniu nowych i pozbywaniu się starych członków. Odeszli Mac i S.S. Captain, wstępując do zachodniej grupy łamiącej programy - Fusion. Na miejsce S.S.'a doszedł z BBS'em **Easy Rider,** a także **PSV** (swapper, decoder), **Smuggler** (koder) i **Backfire** (swapper). Wielu uważa, iż wraz z malejącą produktywnością grupy, a także z odejściem dwóch z trzech najbardziej znanych członków - czasy świetności Katharsis minęły. Przyszłość pokaże czy stać ich jeszcze na odbudowanie dawnej potęgi.

Na podstawie informacji od Mr.Root'a opracował:

Zbigniew Piotrowicz



LISTY DO I OD REDAKCJI

dokończenie ze str. 17

cy problem: mimo iż posiadam kompilator, nie potrafię sprawić, aby program uruchamiał się z CLI lub bezpośrednio z dysku, a nie spod okna edytora. Mimo wielokrotnych prób kompilacji, nie udało mi się wczytać napisanego przeze mnie programu. (...)

Sławomir Ostrowski

Przyczyn tego zjawiska może być kilka (zakładając, że udało się nam już skompilować program):

1.1. Brak biblioteki diskfont.library w katalogu Libs z dysku na którym nagrany jest skompilowany plik.

1.2. W samym kompilerze, w menu "Compiler Program Setup", opcja "Create default screen" jest ustawiona na "No". W tym przypadku, Amos nie stworzy automatycznie podstawowego ekranu dla programu, a pierwsza komenda odnoszące się do niego, spowoduje błąd i wyjście z programu. Najlepiej ustawić tę opcję na "Yes", zaś pozostałe na "No".

1.3. Uszkodzony plik Compiler.AMOS, lub jego nieumiejętna przeróbka przez niepowołaną osobę.

Jeżeli kompilator nie ma ochoty kompilować w ogóle, to najczęściej spowodowane jest to:

2.1. Brakiem wszyskich plików niezbędnych dla kompilera (m.in Compiler.library, A_comp.lib).

2.2 Nieprzetestowanym programem - komunikat "program not tested !!!".

2.3. Patrz punkt 1.3.

c. d. na stronie 36

Fast,Faster... the Fastest (RAM)

Niedawno miałem okazję przeczytać w jednym z czasopism komputerowych test (?) rozszerzenia pamięci o nazwie *Megamix* produkcji niemieckiej firmy *Tri-State*. W ramach uzupełnienia tematu postanowiliśmy przyjrzeć się z bliska trzem innym rozszerzeniom pamięci typu Fast-Ram (ten prawdziwy - dołączany z boku obudowy) dla A-500. Mamy zatem w programie urządzenie o nazwie *MEGA-RAM* produkcji polskiej (!) spółki *Elsat* (pewnie ELektronika SATelitarna...). Dla uproszczenia będziemy go dalej nazywać "MR". Następnie urządzenie dalekowschodniej (Tajwan, Singapur, Malezja, Filipiny, Hong-Kong itp.) produkcji markowane znakiem *Golden Image*. Te będziemy nazywać "GI". Jak również produkt firmy *Supra Corporation* o nazwie *Supra RAM 500RX*. Nazwijmy je "RX". Wszystkie trzy urządzenia mają spełniać to samo zadanie: dodać trochę pamięci do naszej "pięćsetki" nie blokując przy tym dostępu do EXPANSION PORT'u, tak abyśmy nie musieli rezygnować np. z twardego dysku czy cartridge'a podłączanego w to samo miejsce. W przestrzeni adresowej wszystkich Amig wyposażonych w procesor 68000, przewidziano miejsce na zainstalowanie do ośmiu megabajtów pamięci typu "FAST".

Pamięć ta jest widziana przez system w obszarze od $200000 do $A00000. Wszystkie prezentowane rozszerzenia zostały wyposażone fabrycznie w układy pamięci RAM o sumarycznej pojemności dwóch megabajtów. W przypadku GI możemy dodać jeszcze 2MB, a w pozostałych przypadkach dwa lub sześć megabajtów.

Zacznijmy zatem od GI. Z daleka dosyć solidnie wyglądająca obudowa tego rozszerzenia traci dosyć sporo po bliższym zapoznaniu się z nią. Jak większość produktów markowanych znakiem GI, również i ten nie zachwyca zbytnią precyzją wykonania. Po dołączeniu do A500 okazuje się, że urządzenie działa (!) (przynajmniej tak sugeruje świecąca kontrolka w obudowie) ale nasz komputer stał się znacznie szerszy.

MAPA PAMIĘCI AMIGI	
$000000	512 KB Chip-RAM
$080000	Chip-RAM c.d. lub odbicie obszaru $000000-$080000
$100000	Chip-RAM c.d. lub odbicie obszaru $000000-$080000
$180000	Chip-RAM c.d. lub odbicie obszaru $080000-$100000
$200000	8MB FAST-RAM
$A00000	układy CIA
$C00000	rozszerzenie pamięci i wczesne A2000 (pseudo-fast)
$C80000	nie wykorzystane lub ew. pseudo-fast c.d.
$DC0000	zegar czasu rzeczywistego A500 i A2000
$DF0000	CUSTOMCHIPS
$E00000	nie wykorzystane
$E80000	obszar rozszerzeń sprzętowych (SLOTS)
$F00000	moduły ROM
$F80000	Kickstart (512 kB) lub odbicie Kickstartu 256 kB
$FC0000	256 kB Kickstart ROM

Rozmiary (szerokość) obudowy GI przekraczają normy przeznaczone dla twardych dysków połączonych w jednej obudowie z kontrolerem (sterownikiem) SCSI i miejscem na osiem megabajtów RAM'u... Tego typu szafowanie miejscem na stole ewentualnego użytkownika na pewno nie odpowiada dzisiejszym standardom. Strach myśleć co się będzie działo gdy zechcemy do naszej A500 podłączyć jeszcze twardy dysk i cartridge. Spróbujmy zatem! Odłączamy ponownie GI od komputera (oczywiście nie zapominając wcześniej go wyłączyć) i odszukujemy sprytną przykrywkę z boku obudowy.

Przykrywka ta z niewiadomych powodów (może ze strachu, że ktoś będzie chciał coś tam podłączyć) została zamaskowana specjalną nalepką. Po usunięciu tejże (nalepki) należałoby w/w przykrywkę usunąć. Współczuję wszystkim, którzy zasugerują się wytłoczoną strzałką i będą próbowali wyjąć przykrywkę we wskazywanym przez nią kierunku. Jeżeli w porę nie zrezygnujemy, to próby takie skończą się prawdopodobnie uszkodzeniem obudowy. Jeżeli natomiast uda nam się wydedukować właściwy kierunek wyjmowania przykrywki, to efekt jest natychmiastowy.

Po tym pierwszym sukcesie łapiemy co tam posiadamy np. twardy dysk i próbujemy dołączyć go do świeżo odkrytej szczeliny w obudowie GI. Tu okazuje się, że to jeszcze nie koniec niespodzianek jakie zgotowali nam producenci z dalekiego wschodu. Co tym razem? Okazuje się, że krawędź płyty, do której chcieliśmy się podłączyć znajduje się tak głę-

boko wewnątrz obudowy, że mimo szczerych chęci nie uda nam się naszych dwóch urządzeń połączyć ze sobą.

Czy jest na to rada? Jest! Z czeluści pudełka po GI wydobywamy... przedłużacz do złącza krawędziowego. Taki sobie mały kawałek płytki drukowanej a ile radości! Swoją drogą ktoś musiał nieźle pogłówkować, żeby coś takiego wymyśleć. Zawsze to 86 styków więcej do brudzenia się... No, ale dosyć złośliwości. Podłączamy twardy dysk i... Amiga kwituje nasze wysiłki mrugającą zalotnie lampką POWER... Widać producent tyle wysiłku włożył w wymyślanie rozmaitych atrakcji dla ewentualnego użytkownika, że zapomniał o tym, że nie każdy już dzisiaj lubi dyskoteki i mrugające lampki nie zawsze będą mile widziane. Znając już z doświadczenia takie przypadki postanowiliśmy spróbować podłączyć nie twardy dysk do GI a odwrotnie.

Udało nam się od razu (nawet przedłużacz nie był potrzebny). Problem tylko taki, że efekty uzyskaliśmy podobne jak przy poprzedniej próbie (tej z przedłużaczem). Skoro nie udało się z twardym dyskiem, to może jakiś cartridge? Proszę bardzo! Ale również nie działa... To może chociaż zobaczmy czy to w ogóle działa! Po wyszukaniu gdzieś z zapasów dyskietki systemowej "Workbench" uruchamiamy Amigę z podłączonym do niej tylko rozszerzeniem Golden Image.

O dziwo Workbench ładuje się poprawnie i nawet pokazuje nam na górnej krawędzi ekranu, że mamy dwa megabajty "other mem" (OS2.0). Wszystko wydaje się działać poprawnie. Niestety na sprawdzenie prawidłowego działania poważniejszych programów nie pozwolił nam brak możliwości uruchomienia twardego dysku.

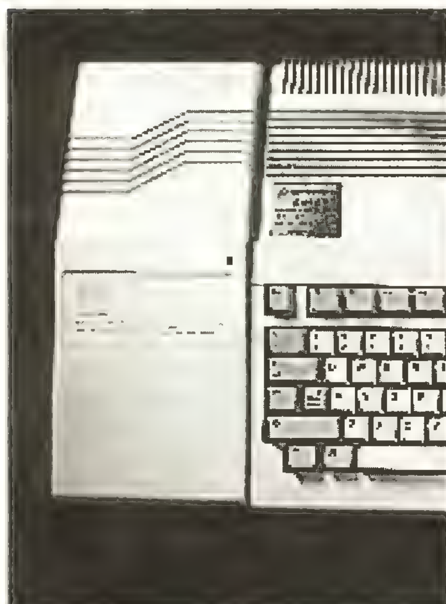
Jak widać na powyższym przykładzie AUTOCONFIG * jest dla niektórych producentów zbyt skompplikowanym rozwiązaniem. Spróbujmy zatem zobaczyć jak to jest wykonane w środku. Po demontażu obudowy oczom naszym ukazuje się płyta drukowana zaopatrzona w 2MB RAM'u w jednomegabitowych układach typu DIP. Ewentualne dodatkowe dwa megabajty możemy wstawić korzystając z modułów typu SIMM.

Wykonanie płyty nie pozostawia wiele do życzenia. Dobra jakościowo i technologicznie płyta nie idzie niestety w parze z właściwym działaniem.

Drugim kandydatem do oceny jest w dniu dzisiejszym produkt warszawskiej firmy "Elsat". Urządzenie o dumnej nazwie *Mega-Ram* sugeruje nam od razu całe 1MB. Okazuje się, że jest tam tego RAM'u więcej (2MB). Po wypakowaniu z nienajładniejszego (rzecz gustu) ale za to dość funkcjonalnego opakowania mamy okazję podziwiać cacko rodzimej (!) ponoć produkcji.

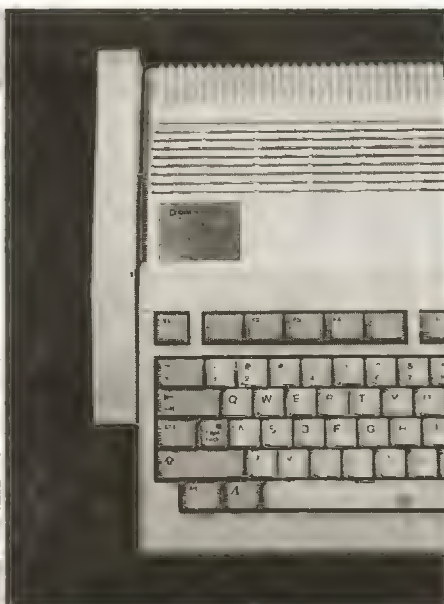
Metalowa obudowa z dużym czarnym napisem, niewiele ustępuje swoją szerokością tej od GI. Za to jest trochę krótsza. Jak możemy zobaczyć na zdjęciu, komuś kto projektował obudowę od MR, chyba ciągle "obsmykiwała" się ręka. Efektem tego jest fakt, że ani przednia, ani tylna krawędź nie jest dopasowana do odpowiednich krawędzi obudowy Amigi. Najbliższa ideału była znajdująca się mniej więcej w 1/3 długości MR krawędź załamania. Jednak i tu brakuje ładnych paru milimetrów aby pasowała ona do tej na A-500. Mimo tego, że (jak wspominałem) estetyka to rzecz gustu, to muszę stwierdzić, że mnie osobiście razi coś takiego. Odłóżmy jednak na bok te dywagacje natury estetyczno-formalnej i przejdźmy do konkretów. Wewnątrz opakowania oprócz samego MR znajduje się jeszcze dyskietka i całkiem przyzwoita pod względem poligraficznym instrukcja obsługi. Również treść instrukcji była dla mnie miłym zaskoczeniem.

Ktoś przyzwyczajony do fatalnych zazwyczaj pod każdym względem instrukcji dołączanych (lub nie) do produktów naszej polskiej


GOLDEN IMAGE - FAST RAM



"ELSAT" - MEGA RAM


SUPRA RAM 500RX

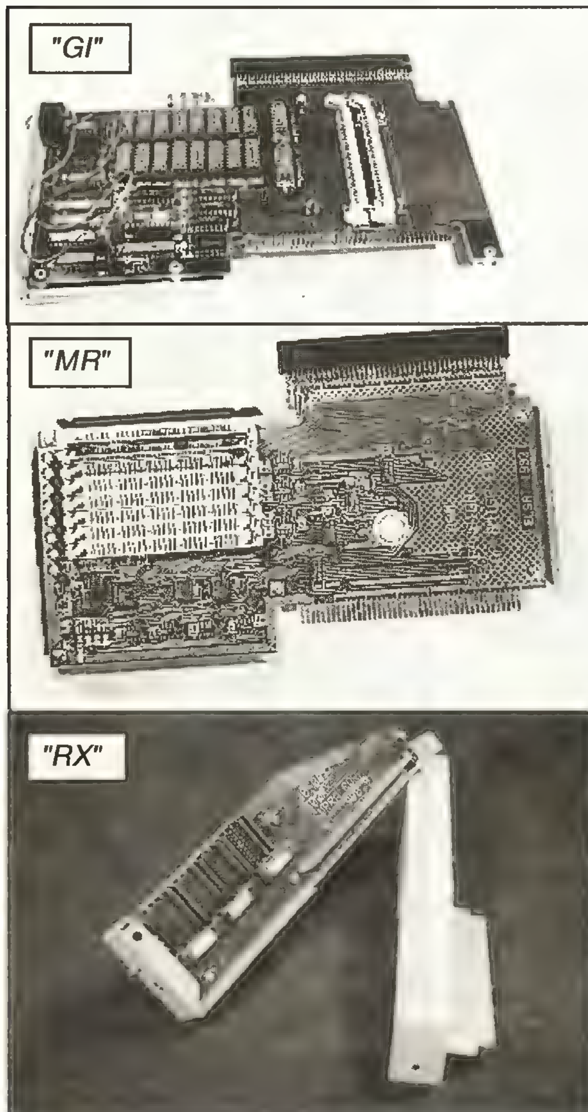
przedsiębiorczości, może dostać w pierwszej chwili ataku czkawki ze zdziwienia. Instrukcja w sposób przejrzysty i rzeczowy tłumaczy wszystko co istotne z punktu widzenia użytkownika MR.

Równie silnym zaskoczeniem jest zawartość dołączonej dyskietki. Co prawda brak w mojej szufladce systemowej "FONTS:" fontu o nazwie pol-top powoduje natychmiastowe odwiedziny GURU, ale po resecie boot'ujemy Amigę z dyskietki i... mamy obrazkową instrukcję obsługi.

Dobrej jakości, digitalizowane obrazki prowadzą nas za rączkę przy podłączaniu nowego nabytku. Co prawda ja osobiście nie położyłbym do zdjęć (digitalizacji) Amigi na ławie "na wysoki połysk" przykrytej obrusikiem w szlaczki i wzorki (eh, estetyka) bo to trochę zaciemnia obraz ale poza tym nie ma się do czego przyczepić.

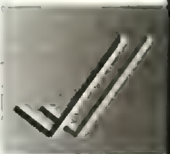
No, może jedynie do tego, że w momencie gdy napis głosi "wyłączyć komputer" to obrazek pokazuje zasilacz z przełącznikiem w pozycji "I". Co by jednak nie mówić, pomysł doskonały i wykonanie całkiem przyzwoite. Przed podłączeniem postanowiliśmy od razu zdjąć przykrywkę przesłaniającą dostęp do złącza krawędziowego z boku. Aby tego dokonać nie potrzebowaliśmy stosować żadnych trików. Wystarczył zwykły, płaski (?!) wkrętak i odkręcenie dwóch mosiężnych śrubek od spodu obudowy. Również dołączenie twardego dysku nie przysporzyło nam problemów i nie wymagało stosowania np. przedłużacza...

Po podłączeniu wszystkiego okazało się, że Amiga... działa! System startuje z twardego dysku bez problemów. Żaden program nie zachowuje się nienormalnie. Słowem: wszystko OK! No, może prawie wszystko. Występują pewne problemy w pracy z cartridge'ami typu X-Power czy AR. W pewnych momentach (po aktywacji freezera) następuje po prostu całkowite "zawieszenie" komputera.Sprawdzamy jeszcze inne kombinacje (kolejność) podłączeń.



Każde urządzenie kompatybilne z Amigowym AUTOCONFIG'iem powinno działać niezależnie od tego czy melduje się jako pierwsze, czy jako drugie, piąte itp. Tu niestety drobne ostudzenie naszych zapałów. Okazuje się, że MR działa poprawnie tylko wtedy, gdy jest podłączony do komputera jako pierwszy. Każda inna kombinacja powoduje, że system się nie uruchamia. Na zakończenie naszego testu należałoby jeszcze zajrzeć do środka. Odkręcamy kolejne, mosiężne, żółte (znowu ta estetyka...) śrubki z płaskim nacięciem.

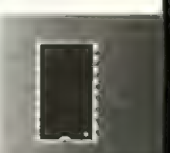
Tak na marginesie, to pomijając już kolor śrubek, który niezbyt mi pasuje do dosyć wiernie odtworzonego "Amigowego" koloru obudowy, to trzeba pamiętać o tym, że świat nie dla kaprysu odszedł już niemal zupełnie od płaskiego nacięcia na łebkach śrub i zastąpił je innymi (np. krzyżowym Davisa).



Po odkręceniu śrubek, kolejna niespodzianka. Cała płyta rozszerzenia wykonana jest w technologii SMT. Po raz pierwszy w polskim produkcie spotykamy się z czymś takim. Co prawda sposób wykonania samej płyty łudząco przypomina produkcję rodem z Hong-Kongu z połowy lat osiemdziesiątych, a nigdzie (przynajmniej z wierzchu) nie jest napisane gdzie właściwie płyty są wykonywane, to jednak napis głosi "ELSAT 1992".



Niezależnie od tego, skąd faktycznie pochodzi produkcja ELSAT'u, należą się wam panowie producenci spore słowa uznania. W rozszerzeniu MR zastosowano pamięć zawartą w jednomegabajtowych modułach typu SIMM. Fabrycznie mamy zainstalowane dwa takie moduły. Dostawienie kolejnych megabajtów wymaga przestawienia odpowiednich zworek (jumper'ów). Tu również miła ciekawostka.



Na płycie zaraz obok właściwych styków mamy namalowane wszystkie możliwe kobinacje zworek dla poszczególnych konfiguracji (2, 4, 8MB). Niby drobiazg, ale jak użytkownik wpadnie na pomysł dostawienia dodatkowych modułów np. po dwuletniej eksploatacji, to może się okazać, że instrukcja obsługi już dawno została pogryziona przez psa, przeczytana przez dwuletnią siostrę lub przydarzyła jej się jakaś inna tragedia. W tej sytuacji trudno o lepsze rozwiązanie niż gotowa ściągawka na płycie.

Na podsumowanie naszego testu MR, należałoby powiedzieć, że ponieważ mamy możliwość porów-

nań z produktami najlepszych firm światowych, to stawiamy bardzo wysokie wymagania testowanym urządzeniom. W takiej sytuacji spodziewaliśmy się znacznie gorszych rezultatów. Pozytywne niespodzianki na każdym niemal kroku zmusiły nas jednak do weryfikacji poglądów. Mega-Ram wydaje się być produktem na całkiem przyzwoitym poziomie.

Na koniec jednak zachowaliśmy prawdziwy rodzynek w dziedzinie rozszerzeń pamięci do A500. *Supra RAM 500RX* Tak nazywa się to cacko i jest to naprawdę cacko. W ogóle wszystkie (prawie) urządzenia markowane znakiem "Supra", to synonim jakości. Piszę "prawie" bo miałem możliwość "doczepić" się do szybkiego modemu ostatniej generacji również produkcji firmy "Supra".

Wróćmy jednak do RX'a. Małe, zgrabne pudełeczko z eleganckim (subtelnym) napisem na obudowie potrafi zawrzeć w swoim wnętrzu całe osiem megabajtów RAM'u. Obudowa RX'a dokładnie przylega do obudowy Amigi i jest do niej również doskonale (moim zdaniem) dopasowana pod względem estetycznym.

Jakość wykonania nie pozostawia nic do czego można by mieć zastrzerzenia. Nawet śrubki mocujące dwie części obudowy czy przykrywkę przesłaniającą złącze krawędziowe, są wykonane tak, że po wkręceniu nie wystają ponad płaszczyznę obudowy i są w tym samym co i ona kolorze. Wewnątrz również Hi-Tech. Precyzja pod każdym względem i wszechobecna technologia SMT.

Jako pamięć możemy w przypadku RX'a wykorzystać jedno, lub czteromegabitowe układy typu ZIP. Zależnie od zastosowanego rodzaju możemy uzyskać maksymalnie dwa lub osiem MB. Projektanci pomyśleli również o starych Amigach wyposażonych w niezbyt silne zasilacze. Co prawda rozszerzenie nie pobiera zbyt dużo prądu, ale jeżeli nie chcemy dodatkowo obciążać naszego zasilacza, to z tyłu obudowy mamy gniazdo do którego możemy podłączyć dodatkowe zasilanie zewnętrzne.

Kompatybilność? Żadnych problemów! Niezależnie od kolejności podłączenia urządzeń wszystko działa bez zarzutów.

U nas w celach testowych pracowały w skrajnych sytuacjach następujące urządzenia podłączone jednocześnie w różnej kolejności do jednego Expansion port'u: Kontroler SCSI "Supra 500XP", Kontroler SCSI "Trinology seria CHA", Cartridge "X-Power" no i oczywiście Pamięć "Supra RAM 500RX" - dwie sztuki. Właściwie nie ma już co więcej pisać na temat RX'a.

Znacznie łatwiej się pisze, jeżeli można się doszukać jakichś wad czy niedociągnięć. Tu, w tych kilku zdaniach został wyczerpany cały temat. *Supra RAM 500RX*, to po prostu najlepsze pod każdym względem rozszerzenie pamięci jakie może sobie zafundować posiadacz Amigi 500 (+).

SD!

---

*AUTOCONFIG - unikalny Amigowy system konfiguracji urządzeń dołączanych do systemu. Charakteryzuje się dynamiczną alokacją obszarów pamięci przeznaczonych dla danego urządzenia.

W AUTOCONFIG'u wszystkie karty rozszerzeń sprzętowych zgłaszają się w kolejności podłączenia w momencie włączenia zasilania lub RESET'u.

Produkt: Mega Ram

Producent: ELSAT sp.c. Warszawa

Dostawca: Sklep "Admiral-Comp"

Cena: 2.950 tys. zł. (10/92)

+ dostateczna kompatybilność

+ wysoka jakość wykonania

+ rzeczowa i zrozumiała instrukcja oraz dyskietka instruktażowa

+ stosunkowo niska cena

- duże rozmiary obudowy

- estetyka (a raczej jej brak)

Ocena KEBAB'a (1-6): 4 - dobra

Produkt: Rozszerzenie pamięci

Producent: Golden Image (?)

Dostawca: Sklep "Admiral-Comp"

Cena: 3.800 tys. zł. (10/92)

- duże rozmiary obudowy

- maksimum 4MB RAM

- niska kompatybilność

- wysoka cena

Ocena KEBAB'a (1-6): 2 - mierna

Produkt: Supra RAM 500RX

Producent: Supra Corporation - USA

Dostawca: Sklep "Admiral--Comp"

Cena: 3.800 tys. zł. (10/92)

+ bardzo wysoka kompatybilność

+ bardzo wysoka jakość wykonania

+ małe wymiary obudowy

+ dodatkowe gniazdo zasilania

+ estetyka

Ocena KEBAB'a (1-6): 1 - celująca

AMOS cz.IV

W tym numerze zapoznamy się z możliwościami manipulowania ekranami. Czasami przy zastosowaniu instrukcji, które zaraz będziemy opisywać, zdarzy się, iż nie będą one pracować zgodnie z naszym zamiarem. Recepta na to jest jedna: dokładnie sprawdzić podane po instrukcji parametry. Szczególnie należy zwracać uwagę na ograniczenia sprzętowe Amigi, jak i na poprzednio wykonane instrukcje, które mogły spowodować jeszcze inne warunki.

A oto spis komend :

`Screen Open N,X,Y,K,R` - otwiera ekran o numerze N, szerokości X, wysokości Y, liczbie kolorów K, oraz rozdzielczości R. Wartości X i Y są tylko wymiarami i nie mają nic wspólnego z rozdzielczością. Parametr R podajemy w słowach "Lowres" dla rozdzielczości poziomej 320 lub "Hires" dla rozdzielczości poziomej 640 punktów. Z trybu Interlace w Amosie nie da się korzystać. Przy otwieraniu ekranów wymagana jest podstawowa znajomość możliwości graficznych Amigi, aby czasem nie próbować czegoś takiego:

`Screen Open 1,640,200,4096,Hires` - tyle kolorów w Hi-Resie???

Poprawnym jest natomiast zapis:

`Screen Open 0,888,555,16,Lowres,` chociaż wygląda bardziej "dziwacznie".

`Screen Close N` - zamyka ekran o numerze N.

`Default` - zamyka wszystkie okna, otwierając podstawowy ekran 0.

`Screen Display N` - wyświetla ekran N.

`Screen Display N,X1,Y1` - wyświetla ekran w określonym przez współrzędne sprzętowe miejscu.

Uwaga !!! Współrzędne sprzętowe to zupełnie co innego niż współrzędne ekranu. Punkt (0,0) we współrzędnych sprzętowych to punkt (-129,-45) we współrzędnych ekranowych. Na tę różnicę trzeba brać poprawkę i przeliczać ich wartości do zastosowania w takich komendach jak m.in. powyższa.Do konwersji współrzędnych stosuje się następujące funkcje:

`X Screen(X) i Y Screen(Y)` - przyjmują wartość współrzędnych ekranowych, będących odpowiednikami sprzętowych współrzędnych X i Y.

`X Hard(X) i Y Hard(Y)` -działanie odwrotne do X Screen(X) i Y Screen(Y).

`Screen Clone N` - kopiuje bieżący ekran, tworząc nowy ekran o numerze N.

`Screen N` - powoduje, że bieżącym ekranem będzie ekran N.

`Screen` - funkcja przyjmująca numer bieżącego ekranu.

`Screen To Front N` - "wyciąga na wierzch" ekran o podanym numerze, jeżeli włączony jest tryb Auto View. Komenda użyta bez parametru N, "wyciąga" ekran bieżący.

`Screen To Back N` - chowa ekran N pod inne ekrany. Użyta bez parametru chowa ekran bieżący.

`Screen Hide N` - chowa ekran N (lub bieżący).

`Screen Show N` - pokazuje poprzednio schowany ekran.

`Screen Copy N1 To N2` - kopiuje zawartość ekranu N1 do N2.

`Screen Copy N1,X1,Y1,X2,Y2 To N2,X3,Y3,T` -j.w. ale przenosi tylko część ekranu określoną współrzędnymi. Współrzędne X3 i Y3 oznaczają miejsce na ekranie N2 do którego będzie przeniesiony obraz. Parametr T oznacza tryb kopiowania, jednak warto go pomijać ze względu na małe zastosowanie praktyczne i możliwość otrzymania "badziewi" zamiast grafiki.

`Screen Offset N,X1,Y1` - przesuwa lewy górny róg ekranu N do podanych współrzędnych, tym samym przesuwając cały ekran.

`Screen Base` - adres tablicy używanej przez Amos do obsługi ekranów. Tej funkcji w zasadzie nigdy się

nie stosuje.

Screen Height N i Screen Width N - funkcje przyjmujące wartość odpowiednio wysokości i szerokości ekranu N lub bierzącego.

Screen Colour - funkcja przyjmująca wartość ilości dostępnych kolorów w bierzącym ekranie.

Fade X - ściemnia ekran w ciągu czasu X.

Fade X,$RGB,$RGB,$RGB... - płynnie przechodzi w nową paletę kolorów.

Fade X To E,M - przenosi paletę do ekranu E. X oznacza czas, a M jest szesnastobitową maską mówiącą, które kolory mają być modyfikowane. Np. M=%0000000000001111 zmienia tylko 4 ostatnie kolory.

Colour Back $RGB - zmienia kolor "martwego pola", czyli miejsca, gdzie nie ma ani ramki, ani otworzonego ekranu.

Shift Up X,A,B,T - w ciągu czasu X następuje rotacja palety kolorów. Kolor pierwszy staje się drugim itd. Parametry A i B określają pierwszy i ostatni kolor, zaś T=0 powoduje, że ostatni kolor będzie "gubiony",a przy T=1 wstawiany na pozycję A.

Shift Down X,A,B,T - j.w. ale przeciwny jest kierunek rotacji.

Shift Off - zatrzymuje rotację.

Flash N,"($RGB1,T1)($RGB2,T2)...($RGBn,Tn)" - powoduje miganie koloru N. T1, T2, T3 -to czas w jakim ma być widoczny przyporządkowany im kolor.

Flash On i Flash Off - odpowiednio włączają lub wyłączają miganie.

Scin(X1,Y1) - funkcja przyjmująca wartość numeru ekranu, w którym znajduje się punkt o podanych współrzędnych sprzętowych.

Auto View Off - powoduje, iż nowo otwarty ekran komendą Screen Open, nie będzie automatycznie wyświetlany.

Auto View On - przywraca automatyczne wyświetlanie.

View - wyświetla zmiany wstrzymane rozkazem Auto View Off.

Default Palette K1,K2... - działanie podobne do instrukcji Palette, ale ustawiane kolory są obowiązujące dla wszystkich otwieranych ekranów.

Get Palette N - nadaje bierzącemu ekranowi paletę kolorów z ekranu N.

Load Iff "nazwa" - ładuje grafikę i wyświetla ją na bierzącym ekranie.

Load Iff "nazwa",N - ładuje grafikę na ekran N.

Save Iff "nazwa" - nagrywa ekran na dysk.

Save Iff "nazwa",1 - j.w. ale kompresuje plik.

Dual Playfield N1,N2 -tworzy tzw. dual playfield z ekranów N1 i N2.

Dual playfield polega na tym, iż dwa ekrany wyświetlane są równocześnie na monitorze (kolor 0 jest przezroczysty), zachowując przy tym swą pełną niezależność. Najpierw wyświetlony zostaje ekran N2, poźniej N1 (część grafiki z poprzedniego ekranu zostanie zasłonięta). Ekrany w tym celu używane muszą spełniać trzy ważne warunki :

1. Maksymalna liczba kolorów dla ekranów w Hi-Resie to 4. 2. Maksymalna liczba kolorów dla ekranów w Lo-Resie to 8. 3. Nie można łączyć parami ekranów w różnych rozdzielczościach.

Dual Priority E1,E2 - zmienia priorytet ekranów. Ekran który był tłem dla drugiego ekranu w dual playfield, wynieni się z nim "rolami".

Def Scroll N,X1,Y1 To X2,Y2,X,Y - oznacza pole ograniczone współrzędnymi X1,Y1,X2 i Y2 jako pole o numerze N (max.16), wykorzystywane do operacji przesuwania. Parametry X i Y mówią o odległości w pikselach o jaką ma być przesuwana strefa. Ujemne wartości X i Y definiują scrolling w przeciwnym kierunku.

Scroll N - uruchamia przesuwanie N-tej strefy zgodnie z parametrami z instrukcji Def Scroll.

Appear E1 To E2,X,R - tworzy efektowne pojawianie się ekranu E1 na E2. X określa rodzaj pojawiana się, a R rozmiar przenoszonego fragmentu (R można pominąć). Parametr X należy dobrać doświadczalnie samemu, gdyż niektóre jego wartości mogą spowodować pojawianie się tylko części grafiki. Gdy mamy niechęć do doświadczeń, używamy sprawdzonego X=%00110011.

Zoom E1,X1,Y1,X2,Y2 To E2,X3,Y3,X4,Y4 - powiększa lub zmniejsza prostokąt z ekranu E1 ograniczony podanymi współrzędnymi, umieszczając go na ekranie E2, w podanych współrzędnych X3,Y3,X4 i Y4.

Double Buffer - tworzy logiczny bufor widocznego ekranu. Dzięki temu wszelkie operacje na ekranie będą wykonywane najpierw na ekranie logicznym,

a dopiero później jego zawartość zostanie przeniesiona na ekran fizyczny. Buforowanie ekranu "zjada" pamięć, ale pozwala uniknąć niemiłych efektów mrugania. (Nie chodzi tu oczywiście o Interlace, lecz o inne mruganie !!!)

`Screen Swap N` - wymienia ekran logiczny z fizycznym N, lub z bierzącym przy pominięciu N.

`Physic` - funkcja przybierająca numer ekranu fizycznego.

`Logic` - j.w. ale logicznego.

`Wait Vbl` - czeka do momentu zakończenia wyświetlania ekranu. Chodzi tutaj o sprzętowe przenoszenie ekranu z pamięci na monitor, a nie o takie komendy jak np. Appear.

Amos pozwala nam również na zmianę CopperList'y. Poniższe instrukcje przeznaczone są dla zaawansowanych użytkowników, toteż pozwolę sobie jedynie na ich wymienienie (wyjaśnienie szczegółowych zasad wymagałoby nowego artykułu).

`Copper Off` - wyłącza "amosowską" CopperList'ę.

`Copper On` - włącza ją z powrotem.

`Cop Move adres,liczba` - instrukcja Move.

`Cop Movel adres,liczba` - j.w. ale odnośnie 32-bitowego rejestru.

`Cop Swap` - wymienia utworzoną przez nas CopperList'ę z "amosowską".

`Cop Wait x,y` - instrukcja Wait.

`Cop Reset` - ustawia adres następnej instrukcji Copper'a na początek CopperList'y. Instrukcja przyjmuje wartość adresu CopperList'y.

To już wszystko tym razem. Poznane instrukcje + własnoręcznie narysowana grafika, pozwoli na rozbudowanie i zwiększenie atrakcyjności programu Demko z poprzedniego numeru. Wszystko jest po prostu kwestią umiejętności i pomysłowości programisty.

Zbigniew Piotrowicz

Czemu wolę Implodera?

Amiga znana jest z obfitości wszelkiego rodzaju programów do kompresji, w ogromnej większości tworzonych przez amatorów, a używanych do skracania długości dem i "połamanych" gier. Master-Cruncher, Defjam Packer, Tetra-Packer, Byte-killer, Mega-cruncher to zaledwie pierwsze pozycje listy tych najpopularniejszych. Dobrze sprawdzające się w zastosowaniach dla których je stworzono ...i tylko w nich.

Co ma jednak zrobić użytkownik twardego dysku, który średnio co pięć kliknięć myszą ogląda napis "Volume DH0: is full", lub sfustrowany posiadacz systemu minimum, z uporem maniaka usiłujący "upchać" zbiór najpotrzebniejszych użytków na jednej dyskietce? Wymienione wyżej programy nie zdadzą się tu na wiele, gdyż nie potrafią one prawidłowo dekompresować (kompresować można wszystko) plików relokowalnych, stanowiących gro spośród używanych z AmigaDOSem.

Czym właściwie jest program relokowalny? Otóż jest to taki program, który nie ma na stałe przypisanego konkretnego położenia w pamięci komputera, lecz ładuje się tam, gdzie akurat jest wolne miejsce, po czym jest modyfikowany tak, aby móc się w tym miejscu wykonywać. Rzeczą niezbędną do uruchamiania programów relokowalnych jest system operacyjny wyposażony w mechanizm przydziału/rezerwacji pamięci, umożliwiający określenie gdzie właściwie jest "wolne", a gdzie - "zajęte". Systemy takie pojawiły się powszechnie dopiero na komputerach szesnastobitowych, m.in. nawet w MS-Dosie.

Packerów, które są w stanie prawidłowo załadować, zdekompresować i uruchomić program relokowalny pojawiło się na przestrzeni czasu kilka, przy czym największą popularność zyskały dwa spośród nich - PowerPacker i Turbo Imploder (najnowsze wersje odpowiednio 4.2 i 4.0). Choć pierwszy z nich powszechnie uchodzi za lepszy, głównie ze względu na swoją uniwersalność i większą "przyjazność" dla użytkownika, moje zdanie jest jednak odmienne. Postaram się więc pokrótce odpowiedzieć na tytułowe pytanie.

Pierwszą rzucającą się w oczy różnicą jest szybkość rozpakowywania programów. O ile podczas kompresji prym wiedzie PowerPacker, zgoła inaczej się sprawa ma przy procesie odwrotnym. A kompresuje się tylko raz.

Pomijalne są może minimalnie

mniejsze zyski z kompresji wynikające z innej implementacji tego samego algorytmu, sprowadzające się do kilkuset bajtów różnicy na korzyść Turbo Implodera. Zdecydowanie ważniejszą sprawą jest natomiast bardzo nieekonomiczna gospodarka pamięcią, którą prezentuje nam PowerPacker. Kiedy program nim "scrunchowany" ładowany jest do pamięci, OS automatycznie (w wyniku działania funkcji LoadSeg) jako pierwszy rezerwuje blok dla danych skompresowych. Następnie, już po uruchomieniu, decruncher odwołując się do funkcji AllocMem z biblioteki exec przywłaszcza sobie kolejne obszary pamięci znajdujące się ZA wspomnianymi danymi i do nich depakuje program. Gdy proces dobiega końca, obszar zajęty uprzednio przez LoadSeg jest zwalniany i sterowanie jest przekazywane do właściwego programu.

Zdawać się może, że tak właśnie powinno to wyglądać, niemniej powyższy proces ma co najmniej dwa słabe punkty. Po dekompresji i uruchomieniu programu pozostaje nam pośrodku pamięci wolny obszar o wielkości równej długości spakowanego pliku.

Przy wielokrotnym uruchamianiu tego typu programów postępujący proces defragmentacji pamięci pozostawia nas z RAMem po dziurawionym jak sito, tak więc użytki, próbujące zarezerwować najdłuższy ciągły blok pamięci otrzymają jej zdecydowanie mniej (np. ASM-One, Resource, ADPro).

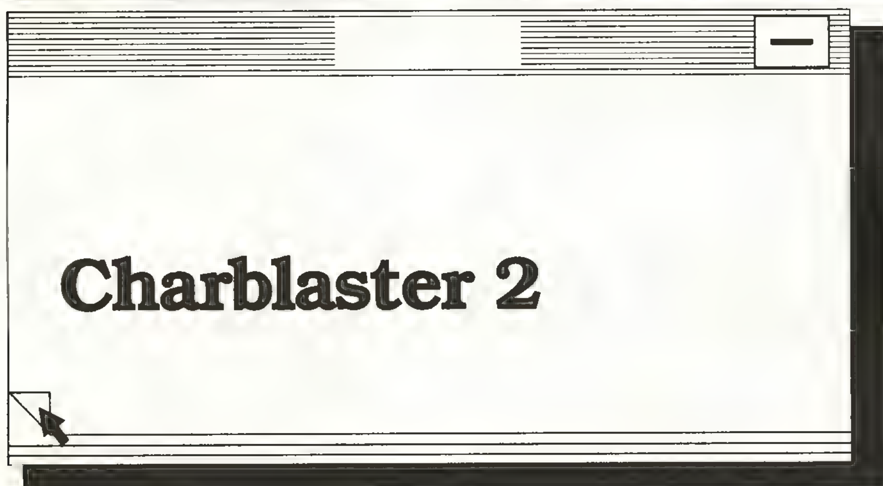
Sprawa druga, to możliwość wystąpienia błędu braku pamięci w momencie wywołania przez decruncher funkcji AllocMem. Po całkowitym załadowaniu programu, jesteśmy informowani o niemożności jego uruchomienia. Starsze wersje PowerPackera wyczyniały w tym miejscu cuda, zwracając sterowanie do systemu bez zwolnienia zarezerwowanych obszarów. Do czego to prowadziło, tłumaczyć chyba nie trzeba.

A Imploder? Poprzez inteligentne zorganizowanie listy hunków w pliku (poszczególnych segmentów, które mogą ładować się w odrębne miejsca), cała niezbędna pamięć jest rezerwowana na samym wstępie procesu ładowania.Jeśli więc program ma się nie zmieścić, o błędzie 103 - braku RAM, dowiemy się natychmiast.

Ta sama lista, zawierając w sobie dodatkowe hunki BSS, wymusza rezerwację odpowiednich obszarów pamięci dla programu jeszcze za pośrednictwem LoadSeg i to PRZED skompresowanymi danymi. Podczas rozpakowywania dane za pośrednictwem specjalnego bufora (również zarezerwowanego jako BSS) są przepisywane pomiędzy tymi obszarami, zaś procedury AllocMem nie wywołuje się wcale. W rezultacie pamięć pozostaje ciągła, co ma szczególne znaczenie dla posiadaczy niezbyt zasobnych w RAM komputerów.

Tak mniej więcej przedstawiają się moje argumenty. Jestem zdania, że choć PowerPacker jest standardem jeśli mowa o kompresji danych (któż z nas nie widział plików z rozszerzeniem .pp), to do pakowania programów wykonywalnych (ang. executables) zdecydowanie bardziej nadaje się Turbo Imploder.

Miłosław "Thorgal" Smyk



Na przełomie lat 1990/1991 na scenie Commodore 64 trwało pewne zamieszanie w związku z kompresorami znacznikowymi (czyli tzw. charpackerami). Nie oznacza to, że powstały one w tym właśnie okresie. Otóż natenczas nowy koder grupy ONEWAY (twórcy m.in. Cruel Cruncher'a) o pseudonimie Zyziphus napisał kompresor znacznikowy, który charakteryzował się bardzo krótką procedurą dekompresującą.

Miała ona bowiem $58 bajtów (88 dziesiętnie), co odbiło się głośnym echem na świecie - inne kompresory znacznikowe miały wówczas około $b0 - $f0.Procedura dekompresji w Zipperze (bo tak się ów $58 kompresor nazywał) była napisana tak szaleńczo, że spędziłem nad nią prawie godzinę zanim ustaliłem, jak "to" pracuje.

Unikając plagiatu postanowiłem napisać własny dekompresor o możliwie jak najmniejszej ilości zużytych bajtów i po kilkunastu próbach "zszedłem" do $66 i... utknąłem! W grudniu 1990 pojawiła się na scenie kolejna produkcja - Beeftrucker australijskiej grupy TERA (autor: Matt). Osiągnięta długość to $56.

W tym czasie zostałem koderem w grupie Science 451 co zbiegło się dość szczęśliwie z przełomem w moich procedurach dekompresujących - udało mi się obniżyć poprzeczkę do $54 a kompresor nazywał się Spoonwalker.

Cisza po tym sukcesie nie trwała długo - Zyziphus napisał Zippera V2; czyli $52, co mnie prawie załamało. Szczęśliwym zbiegiem okoliczności przyszło olśnienie i napisałem własnego $52 a krótko potem $50, co po dziś dzień (z czego jestem, nieskromnie mówiąc, dumny) jest niepobitym rekordem. Cały kompresor został nazwany CharBlaster i naprędce rozesłany po kilku krajach.

Pierwowzór ma w całości 15 bloków na dysku i nie jest do końca pozbawiony błędów (podobno żaden program nie jest), które pojawiają się w najmniej oczekiwanych momentach (szczególnie podczas ładowania programów przeznaczonych do kompresji). Aby móc zamieścić go na łamach Kebaba i jednocześnie wyeliminować dawne błędy, zdecydowałem się napisać go od początku z zachowaniem jedynie oryginalnej procedury dekompresującej.

W obecnej wersji całość CharBlaster'a II ma jedynie 9 bloków długości i pozbawiona jest błędów oryginału (nie jest wykluczone, że posiada dla odmiany jakieś inne; kompresować jednak kompresuje bez zarzutu). Należy go wpisać do pamięci korzystając z programu KOREKTOR lub dowolnego debugger'a.

Słów parę o obsłudze programu. Po jego uruchomieniu (RUN) na ekranie ukazuje się nazwa programu, parę dodatkowych informacji oraz pole ustawiania parametrów. Pierwszym z nich jest argument dla JUMP, czyli adres, pod jaki ma oddać sterowanie dekompresor po skończeniu dekompresji. Następnym parametrem jest $01 VALUE. Tu do modyfikacji mamy tylko młodszą połówkę bajtu, który po dekompresji znajdzie się w komórce $01, odpowiadającej za aktualną konfiguracje pamięci (standardowo $37).

SEI/CLI jest wybierane przy pomocy liter S lub C i oznacza nakaz wyłączenia (SEI) lub włączenia (CLI) przerwań po skończeniu dekompresji a przed wykonaniem skoku pod adres zdefiniowany w JUMP. SAVE NAME to nazwa pod jaką nagrany będzie na dysk gotowy, skompresowany program z dodaną automatycznie procedurą dekompresującą.

Po potwierdzeniu prawidłowości podanych informacji wkładamy dyskietkę z programami do dekompresji i wciskamy SPACE. Następuje wczytanie katalogu (tylko pliki z oznaczeniem PRG) i możemy przystąpić do wskazywania programowi, które zbiory mają być wczytane do pamięci i poddane kompresji. Maksymalna ilość zbiorów do kompresji - 16.

Kolejność ich wczytywania do pamięci jest taka sama jak kolejność ich zaznaczania. Wciśnięcie RETURN na nazwie uprzednio zaznaczonego zbioru pozwala na zmianę adresu jego ładowania.

Należy jednak pamiętać, że kompresor pracuje nad danymi, które znajdują się pomiędzy adresami $07E0 i $FFFF. Podanie adresu niższego niż $07E0 spowoduje zasygnalizowanie błędu i konieczność ponownego wyboru. Wczytywanie zbiorów i ich kompresja zaczyna się po wciśnięciu SPACE. Nagrany zbiór dekompresuje się samoczynnie po podaniu dyrektywy RUN. Uwaga! Dekompresor modyfikuje do własnych potrzeb adresy od $01 do $7F. W przypadku kompresji np. programów w BASIC - patrz: BASIC-STARTER.

Paweł "POLONUS" Sołtysiński

Mapa pamięci c.d.

07E DSKSYNC Z - P

Rejestr synchronizacji odczytu z dysku. Transmisja danych w systemie MFM rozpocznie się gdy w buforze znajdzie się taka wartość jaką uprzednio wpisano do tego rejestru.

080 COP1LCH Z + A
082 COP1LCL Z + A

Wskaźnik do programu Copper'a czyli tzw. Copper List'u. Para tych rejestrów powinna wskazywać adres pod którym znajduje się Copper List. Oczywiście wspomniany adres powinien mieć wartość parzystą i wskazującą pamięć CHIP.

Jak już wspominałem w jednym z poprzednich odcinków cyklu, Copper List jest primitywnym programem, który wykonywany jest przez koprocesor graficzny podczas każdorazowego odświeżania ekranu, czyli co każdą ramkę (w zależności od systemu co 1/50 lub 1/60 sekundy).

Program ten szczegółowo opisuje w jaki sposób ma wyglądać obraz uzyskany na ekranie. Muszą więc znaleźć się tam informacje o kodach kolorów, ich ilości, rozdzielczości ekranu, jego wielkości, ewentualnych sprite'ach, itp... Budując własny Copper List mamy do dyspozycji trzy rodzaje rozkazów:

MOVE - rozkaz przesłania danych. Umożliwia on modyfikację większości rejestrów sprzętowych komputera właśnie za pośrednictwem Coppera. Dzięki temu mamy możliwość na przykład zmieniać kolory ekranu nie tylko przy pomocy procesora głównego, lecz również Coppera.

Instrukcja MOVE (jak każda instrukcja Coppera) jest wielkości długiego słowa, przy czym pierwsze słowo jest numerem rejestru, który chcemy zmodyfikować (proszę zauważyć, że wartość ta musi być parzysta, gdyż wszystkie rejestry leżą pod adresami parzystymi), natomiast drugie słowo jest wartością jaką do danego rejestru chce-

my wpisać (odnośnie tej wartości nie ma żadnych ograniczeń - może ona być zarówno parzysta jak i nieparzysta.)

Przykłady użycia instrukcji MOVE:

dc.l $01800FFF - instrukcja ta umieszcza w rejestrze $0180 wartość $0FFF. Jako że rejestr $0180 odpowiedzialny jest za kolor tła, a wartość $0FFF jest kodem koloru białego, wykonanie tej instrukcji spowoduje zmienienie koloru tła na biały. Analogiczny efekt możemy uzyskać, jeśli wykonamy procesorem głównym instrukcję:

```
move.w #$0FFF,$DFF180
```

Istnieje jednak drobna różnica pomiędzy podaną instrukcją Coppera, a przytoczoną instrukcją asemblera, przejawiająca się nie tyle w wyniku ich wykonania, co w sposobie wykonania. Otóż jeżeli kod $01800FFF będzie jedną z instrukcji naszego Copper'a to jej wykonanie będzie powtarzane co 1/50 sekundy (w systemie PAL), czyli przy każdym odświeżaniu ekranu, natomiast instrukcja asemblera zostanie wykonana tylko raz, o ile oczywiści nie będzie ona umieszczona w specjalnie zorganizowanej pętli. Inne przykłady:

dc.l $0180000F - zapisanie w rejestrze $0180 wartości $000F (kolor niebieski)

dc.l $009A7FFF - zapisanie w rejestrze $009A wartości $7FFF, oraz odpowiednik asemblerowy:

```
move.w #$7FFF,$DFF09A
```

Tutaj chciałbym przypomnieć, a tym co jeszcze tego nie wiedzieli wyjaśnić, że rejestry, które omawiamy możemy modyfikować procesorem po dodaniu do nich adresu bazowego tzw. CUSTOM o wartości $DFF000

WAIT - to drugi obok MOVE najczęsciej używany rozkaz Coppera, służący do wstrzymywania pracy koprocesora aż wyświetlanie ekranu dojdzie do zadanej pozycji. Składnia rozkazu WAIT jest następująca:

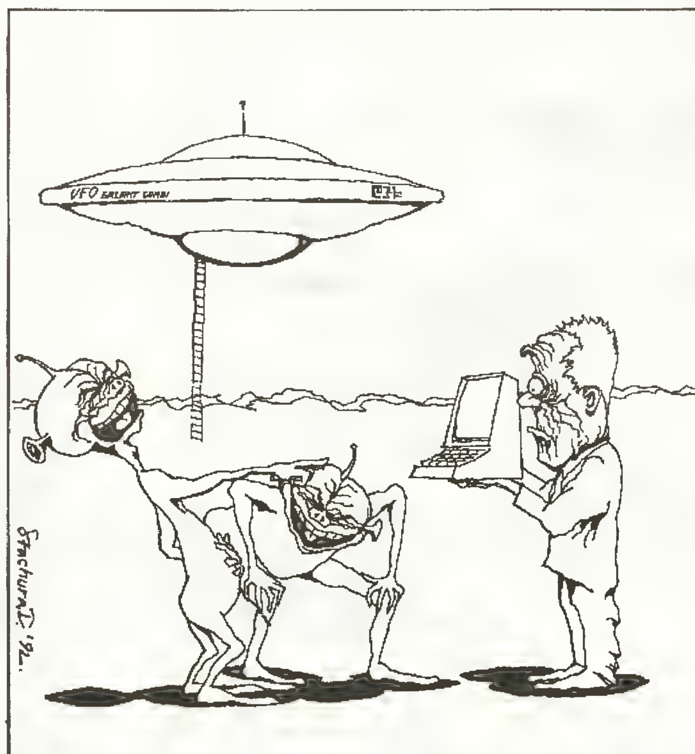
Starsze słowo: bity 15-8 określają pionową linię (numer rastra) na którą czekamy. bity 7-1 określają poziomą współrzędną oczekiwania. bit 0 zawsze ustawiony.

Młodsze słcwo: bit 15 - używany w sterowaniu Blitter'em z poziomu Copper'a, normalnie powinien być ustawiony. bity 14-8 bity pozwolenia porównania pozycji pionowej. bity 7-1 bity pozwolenia porównania pozycji poziomej. bit 0 zawsze skasowany.

Wspaniale!!! ...tylko jak TO wykorzystać? Załóżmy, że chcemy na przykład zmienić kolor tła ekranu począwszy od środka $90 linii na biały. Należy w tym celu wykonać dwia rozkazy Copper'a. Pierwszy będzie rozkazem WAIT oczekującym aż wyświetlanie ekranu dojdzie do środka wspomnianej linii, natomiast drugi rozkaz będzie znaną nam już komendą MOVE przesyłającą do rejestru $0180 wartość $0FFF. Oto jak będzie wyglądać ten fragment Copper'a:

dc.l $9071FFFE - czekanie na linię $90 i pozycję $71 w tej linii

dc.l $01800FFF - ustawienie białego koloru



Myślę, że po tym trywialnym przykładzie, mechanizm oczekiwania na zadaną pozycję powinien być zrozumiały. Wyjaśnienia może wymagać jedynie znaczenie młodszego słowa komendy WAIT, czyli tajemniczych bitów pozwolenia porównania. Dają one programiście możliwość podania współrzędnych oczekiwania w sposób niejawny.

Cóż to oznacza? Biorąc pod uwagę, że Copper może wykonywać pętle, zachodzi czasem potrzeba zapisania instrukcji WAIT bez podawania na przykład pozycji pionowej (chcemy czekać na daną pozycję poziomą w każdej linii). Należy wtedy zamaskować bity odpowiadające pozycji pionowej, po prostu je kasując.

Przykład:

dc.l $007180FE - oczekuje na pozycję $71 w najbliższej linii. Bity 14-8 są skasowane, co powoduje że pozycja pionowa jest ignorowana.

Ustawienie młodszego słowa instrukcji WAIT na wartość $FFFE (jak w pierwszym przykładzie) oznacza, że nic nie będzie ignorowane i koprocesor będzie oczekiwał na pozycję zadaną w starszym słowie. Należy tu zaznaczyć iż stosowanie wszelakich trick'ów z maskami i pętlami jest stosunkowo czasochłonne i należy ich unikać, co niemal zawsze jest możliwe.

Poza standardowymi instrukcjami WAIT oczekującymi na zadane pozycje istnieją dwie posiadające odrębne znaczenie. Pierwszą z nich jest komenda o kodzie:

```
dc.l $FFFFFFFE
```

Oznacza ona zakończenie CopperListu i każdy program koprocesora powinien się taką sek-

wencją kończyć. Drugą z nich jest:

```
dc.l $FFDFFFFE
```

Instrukcja ta czeka aż wyświetlanie ekranu dojdzie do końca 255 linii. Po jej wykonaniu wszystkie wykonania instrukcji WAIT będą się odnosić do dolnej części ekranu, czyli numerów linii $100 - $138. Rozpatrzmy sekwencję rozkazów:

dc.l $FFDFFFFE ;czekamy na koniec linii $FF

dc.l $0201FFFE ;czekamy na 2 linię dolnej części (czyli linię $102)

dc.l $0901FFFE ;czekamy na 7 linię dolnej części (czyli linię $109)

dc.l $FFFFFFFE ;kończymy Copper List.

SKIP - to ostatni z rozkazów rozpoznawanych przez Copper'a. Swoim działaniem powoduje przeskoczenie następującej po nim instrukcji, jeżeli pozycja aktualnie wyświetlanego punktu jest równa lub większa od zadanych współrzędnych. Rozkaz SKIP posiada następującą postać:

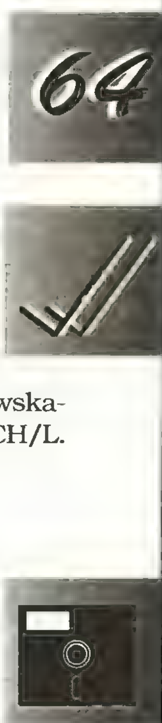
Starsze słowo: bity 15-8 określają pionową linię z którą porównujemy. bity 7-1 określają poziomą pozycję z którą porównujemy. bit 0 zawsze ustawiony.

Młodsze słowo: bit 15 - używany w sterowaniu Blitter'em z poziomu Copper'a, normalnie powinien być ustawiony. bity 14-8 bity pozwolenia porównania pozycji pionowej. bity 7-1 bity pozwolenia porównania pozycji poziomej. bit 0 zawsze ustawiony.

Instrukcja SKIP umożliwia tworzenie zapętlonych Copper List'ów, jednak techniki te są używane tylko sporadycznie, więc na tym zakończę opis tej instrukcji.

084 COP2LCH Z + A
086 COP2LCL Z + A

Rejestry te mają identyczne znaczenie jak COP1LCH/L. Pozwalają one na zdefiniowanie drugiego Copper List'u i programowe (z poziomu procesora głównego lub Copper'a) przełączanie między nimi. Ta możliwość z reguły nie jest wykorzystywana przez programistów.



088 COPJMP1 S A

Rejestr przełączający Copper List. Wpisanie do niego dowolnej wartości spowoduje że od tej pory wykonywany będzie Copper List wskazywany przez rejestr COP1LCH/L.

08A COPJMP2 S A

Patrz COPJMP1, dotyczy Copper List'u wskazywanego przez COP2LCH/L

Krzysztof Kobus

P.S. W dziale Listingi znajduje się przykładowy Copper, który mam nadzieję, ułatwi zrozumienie podanych tu wiadomości.

Kompresory dla Commodore 64

- kilka cennych uwag...

Kiedy tylko uznano, że warto poświęcić sporo miejsca w tym numerze KEBABA kompresorom właśnie, mimowolnie w pamięci mignęły mi rożne przygody z owymi kompresorami w roli głównej. Aby móc bezpiecznie ich używać (tzn. aby czas poświęcony kompresji nie poszedł na marne) potrzebowałem trochę czasu na zdobycie potrzebnego do tego doświadczenia.

Ponieważ kompresory chyba jeszcze nie są w powszechnym użytku, a mamy nadzieję, że po lekturze tego numeru to się zmieni, niejako "od zaraz" potrzebne będą pewne użyteczne informacje choćby po to, by potem nikt nie psioczył "no i co oni tam w tym KEBABIE napisali?! Za nic mi to nie chce działać!". No to zaczynamy od wymienienia kilku prawideł pracy z kompresorami:

1. Rzadko który dekompresor po odtworzeniu programu nie zaśmieca lub wręcz nie niszczy części zmiennych systemowych, wektorów i innych "żywotnych" miejsc w pamięci komputera, ważnych np. dla bezbłędnej pracy interpretera BASIC, procedur obsługi dysków itp.

2. Tylko niektóre z kompresorów przewidują możliwość kompresji programów w BASIC - pozostałe automatycznie startują procedury w języku maszynowym, zakładając, że zdekompresowany właśnie program jest sobie w stanie poradzić z niedogodnościami wymienionymi w punkcie 1 lub, że nie mają one dla niego znaczenia (np. nie korzysta z BASIC, obsługi stacji dysków itp.).

3. Kompresory posiadają ograniczenia jeżeli chodzi o wielkość kompresowanego programu, jego oryginalne położenie w pamięci komputera lub jedno i drugie.

4. Większość kompresorów ja-

ko efekt swojej pracy generuje pojedyncze bloki danych, zawierające na swoim początku linię BASIC z komendą SYS. W ten sposób tak utworzony zbiór można wgrać później do pamięci jako program w języku BASIC i najnormalniej uruchomić go rozkazem RUN. W zasadzie wszystkie gry zostały skompresowane w ten sposób.

5. Z względu na dość wygodną obsługę stacji dysków oraz trudną do przebicia zaletę "czytam bajt po bajcie choćby i co pół godziny, jeżeli mi to odpowiada" generalnie nie ma kompresorów dla użytkowników magnetofonów, a te, które są, nie posiadają zbyt wielkich możliwości.

Po uruchomieniu dowolnego kompresora zadaje on nam kilka pytań, odpowiadając na które ustalamy odpowiednio następujące rzeczy:

LOAD NAME - tu podajemy nazwę programu do załadowania i skompresowania;

SAVE NAME - to będzie tytuł programu wygenerowanego przez kompresor;

START ADDRESS lub JUMP, JUMP TO etc. - podajemy w systemie szesnastkowym adres pod jaki ma "skoczyć" dekompresor po zakończeniu odtwarzania programu (np. $1000 itp.);

$01 VALUE - to wartość komórki $01, która zostanie ustawiona w ten sposób po zakończeniu dekompresji. Odpowiednie jej ustawianie zawiaduje konfiguracją pamięci. Oto lista częściej ustawianych wartości:

$37 - włączone wszystkie pamięci ROM tak, jak po włączeniu komputera;

$36 - tak jak wyżej, za wyjątkiem odłączenia pamięci ROM zawierającej interpreter języka BASIC;

$35 - odłączone wszystko oprócz VICa, SIDa, portów i atrybutów ($D000-$E000);

$34 - odłączone wszystkie ROMy - komputer jest ciemny i głupi...

SEI/CLI - wybranie opcji SEI spowoduje uruchomienie skompresowanego programu z wyłączonymi przerwaniami IRQ; CLI - z włączonymi;

oraz w niektórych dodatkowo:

SCRAMBLE (Y/N)? lub DO YOU WANT TO CODE IT? - oznaczają propozycje "zakodowania" programu skompresowanego tak, aby przed jego uruchomieniem nie można było odczytać np. napisów;

SPEED - w niektórych kompresorach sekwencyjno-bitowych jest to parametr umownie ustalający wielkość każdorazowo przeszukiwanej pamięci w poszukiwaniu nadającej się do kompresji sekwencji bajtów.

Zasada jest prosta: im większa liczba SPEED tym większy obszar przeszukiwania pamięci a co za tym idzie, lepsza kompresja (wydajniejsza) oraz, niestety, dłuższy czas kompresji.

RESTORE $2D/$2E lub $AE/$AF - również w kompresorach sekwencyjno-bitowych. Kompresory te używane są do kompresowania danych spod innego kompresora - najczęściej charpacker'a. Opcja ta służy do umożliwienia współpracy z dawnymi kompresorami (jako kompresorami wstępnymi), którym "do szczęścia" potrzebne było ustawienie niektórych zmiennych systemowych tak, jak gdyby skompresowany program był dopiero co wgrany do pamięci.

Kompresory II fazy, tzn. sekwencyjno-bitowe skracają je, stąd potem potrzeba ustawienia tych wektorów na potrzeby uruchomienia zdekompresowanego właśnie pliku ("staremu" kompresorowi będzie się "wydawało", że skompresowany nim program został przed chwilą załadowany.

A teraz krótki przegląd używanych przeze mnie kompresorów:

Kompresja wstępna (I faza):

BABY-GANG PACKER - stary, ale jary. Może skompresować tylko jeden plik na raz (tzn. program do kompresji musi się znajdować w postaci jednego pliku). Jak na kompresor wstępny - bardzo dobre wyniki kompresji (zwłaszcza tekstów - szczególnie polecam autorom magazynów dyskowych). Program kompresowany może być już od adresu $0199 wzwyż!

Pewną niedogodnością jest wolne ładowanie, co w przypadku kompresora dla pojedynczych zbiorów jest nieco niewygodne. Kompresuje programy w BASIC (pozostawiamy JUMP:$A659). Dodatkowo można umieścić w dekompresorze tekst, który będzie wyświetlony na ekranie podczas dekompresji.

CROSS LINKER - różne wersje od 1.5 do 3.0 w zasadzie funkcjonalnie niewiele się różnią. Słowo LINKER w nazwie oznacza zdolność do wgrywania kilku części składowych (np. grafiki, muzyki i programu) co eliminuje konieczność uprzedniego tworzenia jednolitego zbioru, jakiego potrzebuje np. BABY-GANG PACKER. Należy jednak pamiętać, że części te nie mogą na siebie "nachodzić", tzn. ich obszary nie mogą mieć części wspólnych, gdyż CROSS LINKER stosuje zasadę "ładuje i podczas ładowania kompresuje" a nie "najpierw ładuje wszystko a potem kompresuje". Do ciekawostek należy możliwość utworzenia zbioru startującego od podanego adresu a nie tylko z poziomu BASIC. Na stałe zainstalowano "przyspieszacze" dla stacji 1541.

NO STACK USING PACKER (NSU-PACKER) - czyli Nieużywający Stosu Packer. Do czego to może być potrzebne? Po prostu specjalny kompresor do specjalnych zastosowań. Jak wiadomo stos leży od adresu $0100 do $0200. Z rożnych względów możemy potrzebować tego obszaru a pozostałe kompresory go nam mogą zniszczyć niejako "niechcący", bowiem stos jest wykorzystywany podczas

wykonywania niektórych rozkazów procesora 6502/6510 (żeby daleko nie szukać - np.JSR).

Napisanie takiego dekompresora wymagało uniknięcia w nim właśnie tych rozkazów - w efekcie do dyspozycji mamy pamięć od $0037 - $FFFF !!! Program do kompresji może składać się z dwóch plików (nazwę ewentualnego drugiego podajemy jako odpowiedź na pytanie LINK WITH?...). Zainstalowano szybki system LOAD/SAVE dla 1541.

CHARBLASTER - najnowszą jego wersję można "wklepać" z tego numeru KEBABA. Też z rodziny LINKER'ów, może załadować PRZED kompresją do 32 części (oryginał) lub 16 (jego KEBABOWA wersja). Zaakcentowanie wyrazu "przed" oznacza, że ładowane części mogą się pokrywać, stąd ważna jest kolejność ich wgrywania do pamięci. Bardzo krótka procedura dekompresująca ($50 bajtów) posiada także inną zaletę - krótki czas dekompresji. Konieczność pozostawienia wolnej pamięci nie pozwoliła na instalację szybkiego systemu ładowania i nagrywania dla stacji 1541.

A teraz kolej na kompresory sekwencyjno-bitowe (II faza):

CRUEL-CRUNCHER V2.0 - najstarsza wersja kompresora, jaką posiadam. Dostała się w moje ręce prosto od Autora - Galleon'a z grupy ONEWAY. Był to wówczas NAJLEPSZY kompresor dla Commodore 64 występujący w formie oddzielnego programu. Był tylko jeden lepszy kompresor - Compress Master - specjalna karta włączana do portu Expansion.

Należy pamiętać o tym, że skompresowane dane nie dadzą się prawidłowo uruchomić z aktywnym modułem Action Replay (kolizja programowa). Wada ta dotyczy tylko i wyłączenie wersji 2.0. Następne są wolne od tego "niedopracowania" (tu myślę o wersjach 2.5 i 3.0) i pamiętać wystarczy tylko o tych kilku sprawach:

a) programy kompresowane MUSZĄ startować od adresu $0801 (a więc od początku standardowej pamięci przeznaczonej dla programów w BASIC - użycie dowolnego kompresora I fazy załatwia ten problem);

b) CruelCruncher nie potrafi skompresować danych po uprzedniej kompresji za pomocą BABY-GANG-PACKER'a. A szkoda...

c) zbiór do kompresji nie może być dłuższy niż 225 bloków (to też zwykle nie jest problemem);

Jedyna "wada" - kompresja zajmuje mu trochę czasu.

FAST CRUEL 2.0 do 3.0 - czyli szybka wersja CruelCruncher'ów opisywanych powyżej. Nie różnią się w zasadzie niczym poza dużą prędkością kompresji okupioną o wiele mniejszą wydajnością (ale i tak lepszą niż inne tego typu).

SPEED-PACKER - dość stary ale za to naprawdę szybki kompresor. Ważna informacja - nie kłóci się z żadnym znanym mi kompresorem I fazy. Charakterystycznym znakiem rozpoznawczym jego pracy jest dźwięk, przepraszam za wyrażenie, "pierdzenia" podczas dekompresji. Maksymalna długość programu do kompresji - 230 bloków.

ABUSE PACKER - kompresor o możliwościach przybliżonych do Fast Cruel'i a nawet na wyższych prędkościach (speed'ach) je przewyższający. Ważne - wymaga zainstalowanego DOS'u, który jest w stanie obsłużyć LOAD i SAVE dla obszarów standardowo maskowanych przez pamięci ROM. Program do kompresji NIE MUSI koniecznie zaczynać się od adresu $0801 jak to miało miejsce w przypadku CruelCruncher'ów. Lista efektów specjalnych jest imponująca.

TIME CRUNCHER - dość stary ale pewny kompresor (podobnie jak Speed Packer nie kłóci się z niczym). Efekty kompresji są zwykle o jakieś 10% gorsze od analogicznych w przypadku CruelCruncher'a.

2000 AD MEGA PACKER - porównywalny z Time Cruncher'em.

I to by było na tyle, jeżeli chodzi o kompresory najczęściej spotykane (wierzę, że takimi właśnie się posługuję). Pozostaje jeszcze cała duża rodzina packer'ów, które dekompresują dane podczas ich wgrywania z dyskietki, ale to już historia na później...

Paweł "POLONUS" Sołtysiński

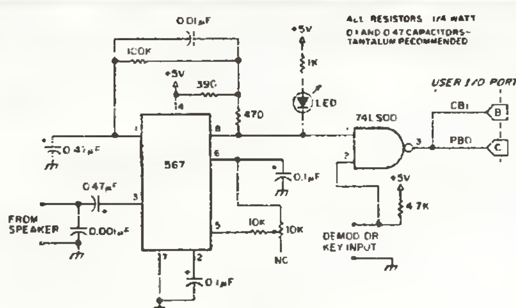


Fig. 2. 73 Morse receive schematic

Zgodnie z obietnicą, przedstawiamy dzisiaj drugą część interface'u do nadawania i odbioru alfabetu Morse'a. Tym razem jest to część odbiorcza. Schemat przytaczamy za "73 Magazine" z czerwca 1984 roku. W następnym numerze zabierzemy się już za Packet Radio.



Sterowanie PowerPackerem z poziomu ARexx'a

Minęły już czasy, gdy słowo "ARexx", wprawiało w wielkie zdziwienie początkujących użytkowników Amigi, lub wśród grona wszelkiej maści "specjalistów" nie bardzo wiedzących o czym mówią, budziło refleksje typu: "To bardzo przydatna rzecz". Dziś wiemy już, że ARexx, jest interpretowanym (choć istnieje również kompilator, niestety narzucający pewne ograniczenia), językiem programowania umożliwiającym komunikowanie się kilku jednocześnie pracujących programów.

W tym krótkim artykuliku nie chciałbym zaczynać kursu programowania w tym języku (krótki przewodnik po ARexx'ie znajdziecie w 5 i 6 numerze Kebaba), jedynie pokazać przykładowe, praktyczne jego wykorzystanie, a że niniejszy numer Kebaba poświęcony jest wszelkiego rodzaju cruncherom, i wszystkiemu co z nimi związane, postanowiłem napisać krótki skrypt w ARexx'ie dla PowerPacker'a.

Co zrobić, aby z niego skorzystać? Przede wszystkim należy zaopatrzyć się w ARexx'a (potrzebny będzie program RexxMast, oraz biblioteka "rexxsyslib.library"), PowerPacker'a v4.0a, oraz CED'a v2.12. Następnie należy uruchomić te trzy programy, wklepać pod CED'a program znajdujący się w dziale listingi i zapisać go na dyskietce.

Gdy załatwimy powyższe formalności możemy przystąpić do uruchomienia programu. W tym celu wybieramy opcję "Send Dos/ARexx command..." z menu "Special:Dos/ARexx interface" i w ukazanym okienku wpisujemy ścieżkę dostępu do naszego ARexx'owego programu, czyli nazwę pod jaką uprzednio zapisaliśmy wpisany program. Jeżeli wszystko do tej pory zrobiliśmy OK, to nasz program powinien się uruchomić i na ekranie pokazać serię okienek informujących nas "co się dzieje".

Jeżeli ktoś nie zorientował się jeszcze co ten program wykonuje, krótki opis. Pierwsza, właściwa linia programu informuje interpreter, że chcemy przekazywać parametry, następnie komendą ADDRESS, przekazujemy sterowanie do ARexx'owego portu CED'a. Wyświetlamy informację użytkownikowi, i znaną już nam komendą przekazujemy kontrolę PP'owi. Teraz jego ekran zostaje "wyrzucony" na wierzch i funkcją DecrunchOnly dekompresujemy plik o podanej nazwie.

Wspomniana funkcja w zmiennej rc (Return Code) zwraca wartość 1 jeśli wskazany plik został zdecrunchowany. W takim przypadku zapisujemy zdekompresowany plik pod nazwą pobraną od użytkownika (to wszystko jedną komendą Save!), pobieramy tę nazwę, a właściwie całą ścieżkę (GetFullName) i podstawiamy pod zmienną "Sciezka". Jeśli zaś plik nie został zdekompresowany wyświetlamy stosowny komunikat. Teraz spowrotem przekazujemy sterowanie do CED'a i wyrzucamy jego ekran na wierzch (CedToFront).

Następnie testując zmienną "Bool" sprawdzamy czy plik został zdekompresowany i w przypadku pozytywnej odpowiedzi otwieramy nowe okno w edytorze i ładujemy w nie uprzednio zapisany plik. W przeciwnym wypadku wypisujemy stosowny komunikat i kończymy wykonywanie programu.

Myślę, że osoby znające choć jeden język programowania, nie powinny mieć kłopotu z analizą programu, tym bardziej, że przy niemal każdej jego linii znajdują się komentarze. W razie ewentualnych kłopotów prosimy o kontakt z redakcją. Jeżeli zaś jesteście autorami programów w tym coraz popularniejszym języku, i chcielibyście przedstawić je Czytelnikom Kebaba to również prosimy o kontakt.

Krzysztof Kobus

UWAGA!!! Z racji tego, że w nazwie "RAM DISK:" znajduje się spacja, prosimy nie zapisywać zdekompresowanych plików na RAM:'ie, gdyż mogą wystąpić kłopoty z instrukcją "Open Sciezka".

Basic Starter 64

Tak się niestety składa, że naj lepsze kompresory powstały z myślą o kompresji programów napisanych w kodzie maszynowym a ściślej rzecz biorąc - zwykle uruchamiają program maszynowy od zadanego adresu zaraz po zakończeniu dekompresji. Jeżeli dodamy do tego fakt częstego niszczenia obszarów tzw. zmiennych systemowych (znajdujących się np. na zero-page) jasnym się staje, że programy napisane w BASIC w takim środowisku mogą sprawić wiele kłopotów w związku z ich poprawnym uruchomieniem.

Szkoda by było jednak sobie odmówić kompresji efektów naszej pracy w języku BASIC, choćby dlatego, że jest on powszechnie wykorzystywany przez bardzo dużą rzeszę użytkowników.

Przypatrzmy się, jak skonfigurowanego środowiska potrzebuje program w języku BASIC, by mógł być prawidłowo uruchomiony. Zacznijmy od zero-page (ogólnie adresy od $00 do $FF):

adres $01 - ustawienie prawidłowej konfiguracji pamięci (wszystkie ROM'y włączone tak, jak po normalnym resecie komputera);

adresy od $2B do $38 - wektory systemowe; zawierają m.in. dane o adresach początku i końca programów w BASIC oraz początkach wolnej pamięci dla zmiennych użytych w programie;

adresy $73 - $8B - procedura GET - pobiera kolejno dane z linii BASIC - bezwarunkowo konieczna do pracy programów w BASIC;

Wektory systemowe ($0300 - $0334) - wektory procedur obsługujących operacje wejścia i wyjścia (w tym także SAVE i LOAD) oraz wektory niektórych funkcji języka BASIC (interpretera, LIST, USR(), itp.).

Zwykle większość z tych obszarów jest podczas dekompresji uszkadzana co raczej nie przeszkadza w pracy programom w języku maszynowym (o ile z nich nie korzystają do swoich celów). Rozwiązaniem jest napisanie krótkiej procedurki w języku maszynowym, która dekompresowana wraz z naszym programem w języku BASIC zostaje przez dekompresor uruchomiona. Jej zadaniem jest naprawienie lub właściwe ustalenie zawartości podanych powyżej obszarów i następnie uruchomienie samego programu w BASIC.

Prezentowany program jest zamieszczony na łamach KEBABA na dwa różne sposoby: pierwszy z nich to wydruk programu w postaci tekstu spod TurboAssemblera (to dla wnikliwych) a drugi to program w BASIC. Uruchomienie tego ostatniego w efekcie powoduje wygenerowanie na dysk samej procedury, która potem może być użyta do uruchomienia programu głównego (w BASIC). Jak wobec tego skompresować napisany przez nas program w języku BASIC.

Do naszych potrzeb wystarczający będzie zamieszczony w tym samym numerze CharBlaster 2. Oto kolejność postępowania:

1. Zapisać na jednym dysku nasz program w BASIC oraz wygenerowaną przez BASIC-STARTER'a procedurę (BAS.START.CODE);

2. Wgrać i uruchomić CharBlaster'a i podać następujące parametry:

JUMP :$C000 $01 VALUE :$37 SEI/CLI :Cli

...następnie nadać dowolną nazwę jako SAVE NAME, potwierdzić prawdziwość podanych danych przez Y i wcisnąć SPACE. Po automatycznym załadowaniu katalogu dyskietki z naszym programem i BAS.START.CODE wybieramy je i wciskamy SPACE. Zostają one załadowane do pamięci, skompresowane i po naszym potwierdzeniu - nagrane na dysk jako jeden program (o nazwie podanej w SAVE NAME).

Jeżeli będziemy nadal zainteresowani dalszą kompresją danych, możemy wybrać odpowiadający nam kompresor sekwencyjno-bitowy (np.Cruel-Cruncher) i podając jako adres startowy start dekompresora (w przypadku CharBlaster'a jest nim $080B) "prasować" całość dalej.

Sam BAS.START.CODE to procedura leżąca pomiędzy adresami $C000 i $C04A, a więc w obszarze zwykle niedostępnym dla programów w BASIC.

Paweł "POLONUS" Sołtysiński

Tanio **sprzedam** C-64 ze stacją dysków.
Szczecin. 346-914

Sprzedam Amigę 500 a 4,5 MB RAM. Cena 9,5 mln.
Mariusz Mentel
ul. Pomorska 94B/91
80-333 Gdańsk

Kupię TOP SECRET nr 1 i 2
Michał Łukaszek
ul. Targowa 50 B
84-300 Lębork

Sprzedam A500 wersja 6A 1MB RAM, bootselector, przełącznik 1MB Chip RAM.
Cena 5 mln.

Sprzedam Amigę 600 + HD 20 MB. Cena 9,5 mln.
Piotr Skirski tel. 483-322
Wrocław

Kupię nietypowe peryferia do C-64, GEOSa sprzedam lub zamienię oprogramowanie.
Sławomir Kędra
ul. Włościańska 16/37
01-710 Warszawa

Oprogramowanie na C-64 (dysk)
Roman Jedlecki
ul. Hetmańska 48/23
58-314 Wałbrzych

Kupię BASIC V.7, inną literaturę,oprogramowanie na C-128 (kaseta)
Bolesław Bukowski
ul. Moniuszki 11/57
11-400 Kętrzyn

Sprzedam C-128, Neptun 156, Action plus V6.0, magnetofon, 2 joysticki, 80 dysków + box, literatura
Michał Czajkowski
ul. Komuny Paryskiej 11/94
85-858 Bydgoszcz

Nawiążę kontakt z grającymi w "FLIMRO'S QUEST"
Michał Orłowski
ul. Żeromskiego 28/5
73-110 Stargard

Tanio **sprzedam** lub wymienię oprogramowanie (ok. 1000 pozycji), duży wybór programów użytkowych.
Hryciuk Piotr
08-201 Niemojki

Sprzedam Action Replay MK III (tanio!), stację dysków 5,25; dyski, box, bootselector. Amigę 500 (1 MB, TOPSTAR, dyski, box) oraz monitor Neptun 156
Andrzej Kaczmarek
ul. Jarochowskiego 101/11
60-248 Poznań
tel. 494-991

A500 1MB, 1084S, gwarancja, akcesoria, 80 dysków, literatura - 12 mln. tel. 532-576
Mirosław Dąbrowski
ul. Piłsudskiego 18
26-110 Skarżysko-Kamienna

Napisałeś jakieś demo, program, muzykę lub grafikę. Chcesz się pochwalić -przyślij a ja rozprowadzę je.
Maciej Lisman
ul. Jugosłowiańska 38/F/9
73-110 Stargard Szczeciński

C-128, monitor Neptun 156, ACTION plus, magnetofon, 2 joysticki, 40 dysków, box.
Cena 6 mln. tel. 638-573
Michał Czajkowski
ul. Komuny Paryskiej 11/94
85-858 Bydgoszcz

Amigę 500 z rozszerzeniem do 1MB, monitor kolorowy, Commodore 1084S, sampler, joystick, literaturę, ok. 400 dysków **sprzedam**. Cena 12 mln.
Michał Mucha
Al. Jedności Narodowej 29/8
Szczecin tel. 223-539

Kupię FINAL II lub FINAL AIII, cena do 100 tys. Oferty na kartkach pocztowych, **kupię** stację 1541 II. oferty z ceną.
Michał Karwowski
ul. M. Dąbrowskiej 1/4
06-300 Przasnysz

Sprzedam C-64 z bogatym osprzętem. Stan b. dobry.
Jakub Jędrzejak tel. 361-992
ul. Wrocławska 32
62-300 Września

Nawiążę kontakt z osobami uczącymi się i programującymi w Amosie w celu wymiany doświadczeń.
Rafał Barbarewicz
ul. Obrońców Poczty G. 4/9
Rzeszów

Programy - Amiga (koperta+ znaczek 2500 zł). Także wymiana.
Henryk Bagiński
B. Głowackiego 2
68-200 Żary

Kupię zielony monitor do C-64 (może być Neptun 156), oraz Action Replay
Krzysztof Marszałek
ul. Boczna 31A/6
44-240 Żory k. Rybnika

Sprzedam C-64 II z magnetofonem, blackbox, joystick, mysz, 20 kaset za 2,5 mln, gwarancja.
Robert Krzymiński
ul. Królowej Jadwigi 34/10
88-100 Inowrocław

Sprzedam stację dysków 1541II (informacje: koperta+-znaczek) C-64 - gry i programy uży tkowe (informacja: koperta +znaczek), (kasety)
Mariusz Listowski
ul. Sobieskiego 17/8
67-200 Słupsk

A 500+ (v.2.04/1.3) 2MB Chip Ram - na gwarancji! Plus monitor 1084S, dyski, literatura.
Krzysztof Jonko
Boh. Warszawy 55/215
Szczecin

Sprzedam gry i programy na C-64 (Informacja: koperta + znaczek)
Paweł Bawiec
ul. Mickiewicza 2
67-100 Nowa Sól

Sprzedam Amigę 500 (1Mb) + joystick + 20 dyskietek.
Cena 6.2 mln. tel.380-11
Marcin Jodełka
ul. Młynarska 10/17
08-110 Siedlce

Sprzedam Amigę 500/1.3 (gwarancja, polska instrukcja) i modulator TV A520
tel. 21-427 do 15.00
Paweł Forczmański
ul. Malczewskiego 35/35
71-612 Szczecin

Poszukuję dem: A PIECE OF SHITIPRS I MORE THAN NOPS /TABOO.
GRUPA ATLANTIC SOFTWARE (C-64) poszukuje kontaktów, muzyków i swapperów Napiszcie!
Tomasz Augustyn
ul. Makarskiego 11/11
49-300 Brzeg

Sprzedaż i **wymiana** programów. Poszukuję HOTROD, SUPER CARS, TECHNOCOP.
Marcin Burkot
ul. Redutowa 2/80
22-400 Zamość

Wymienię gry i programy użytkowe na C-64. Nawiążę kontakt z miłośnikami tego komputera.
Bogdan Bielawiec
ul. Składowa 10/1
61-588 Poznań

Sprzedaż oprogramowania do Amigi, duży wybór, niskie ceny.
Przemysław Mikosz
ul. Buczka 27/12
43-300 Bielsko-Biała
tel. 495-37

Poszukuję opisu programu użytkowego SIMONS+T lub innych programów graficznych.
Tomasz Cieśla
ul. Leśna 45/8
59-300 Lublin

"Tylko o Commodore" **kupię**, cena 10 tys. = koszty przesyłki.
Daniel Żuk
ul. Niedurnego 13/15
44-103 Gliwice

Zamienię komiksy na komputer A500 - mam 399 komiksów.
Grzegorz B.
ul. Wolności 16a/3
22-100 Chełm

Zamienię kamerę QWARC 2x8C.3, projektor "RUŚ", przeglądarkę, koreks na stację 1541 II lub inną.
Aleksander Szubert
ul. 1 Maja 69/37
95-100 Zgierz

Gry, użytki na C-64 (kasety) katalog - znaczek+koperta.
Tomasz Samoraj
Al. Wojska Polskiego 38/25
05-800 Ruszków

Sprzedam C-64, magnetofon, 27 kaset, 12 książek, Black Box V.8, Action plus 7.3, 4 super games. Cena do uzgodnienia.
Robert Wróbel tel. 524-628
Os. 26 Kwietnia 61/21
71-126 Szczecin

Sprzedam Amigę 500, 1MB, 100 dysków, disc box, 2 joysticki za 6.5 mln, sama Amiga - 5.3 mln, monitor 1085s color 10 m-cy gwarancji - 4.1 mln.
Piotr Matuszewski
ul. Wielkopolska 23/39
44-335 Jastrzębie Zdrój

Sprzedam Action Replay I (oryginalny). Oferty z ceną proszę kierować na adres:
Maciej Frontkiewicz
ul. Osiedle 8B/3
49-325 Karłowice

Programy do C-64 (koperta, znaczek). Tanio.
Błażej Strażak
ul. Żorska 20
43-200 Pszczyna

Poszukuję wszelkich programów syntezy mowy. W zamian oferuję gry i programy na C-64 (kaseta).
Bogucki Grzegorz
ul. Żwirki i Wigury 9/4
83-000 Pruszcz Gdański

Tanio **sprzedam** drukarkę EPSON LX-400. Tel. (091)613-815

```
;Copper - przyklad do Mapy Pamieci
;(c) Kebab 1992
;
;Ponizszy Copper wyswietli dwa "bary", a pomiedzy nimi
;fragment pamieci rozpoczynajacy sie od adresu $00000000.
;
;UWAGA!
;Zasemblowany kod umiescic w pamieci CHIP (opcja 'c' przy uru-
;chamianiu wiekszosci assemblerow - AsmOne, Seka).

OldOpenLibrary equ -408

        move.l  #Copper,$dff080        ;ustaw Copper'a.
        move.w  d0,$dff088             ;wystartuj Copper'a.

Loop:

        btst    #6,$bfe001             ;Czekaj na nacisniecie
        bne     Loop                   ;lewego przycisku myszy.


        move.l  4.w,a6                 ;Otwiera biblioteke
        lea     GfxName(pc),a1         ;graficzna, w celu
        jsr     OldOpenLibrary(a6)     ;odtworzenia systemowego
        tst.l   d0                     ;Copper'a.
        beq     End                    ;Znajduje sie on
        move.l  d0,a0                  ;o $26 wzgledem
        move.l  $26(a0),$dff080        ;bazy tej biblioteki.
        move.w  d0,$dff088             ;wystartuj Copper'a
End:
        moveq   #0,d0
        rts

Copper:
        dc.l    $00960020              ;ustawia DMA.

        dc.l    $01080000,$010a0000 ;modulacja ekranu.
        dc.l    $01000000,$01020000 ;rejestry
        dc.l    $01040000           ;kontrolne.

        dc.l    $008e2c50,$00902cc1 ;standardowe
        dc.l    $00920038,$009400d0 ;"bordery".

        dc.l    $3707fffe      ;czekamy na linie $37.
        dc.l    $01800555      ;zmieniamy kolor.
        dc.l    $3807fffe      ;czekamy na linie $38.
        dc.l    $01800aaa      ;zmieniamy kolor.
        dc.l    $3907fffe      ;czekamy na linie $39.
        dc.l    $01800555      ;zmieniamy kolor.
        dc.l    $3A07fffe      ;czekamy na linie $3A.

        dc.l    $01800000,$01800fff ;ustawiamy kolory tla i tekstu.

        dc.l    $00e00000      ;ustawiamy adres pamieci ekranu na
        dc.l    $00e20000      ;wartosc $00000000.
        dc.l    $4007fffe      ;czekamy na linie $40.
        dc.l    $01009000      ;wlaczamy 1 Bit-Plane (dwa kolory)
                               ;w trybie Med-Res.

        dc.l    $8607fffe      ;czekamy na linie $86.
        dc.l    $01000000      ;wylaczamy ekran.

        dc.l    $8707fffe      ;czekamy na linie $87.
        dc.l    $01800555      ;zmieniamy kolor.
        dc.l    $8807fffe      ;czekamy na linie $88.
        dc.l    $01800aaa      ;zmieniamy kolor.
        dc.l    $8907fffe      ;czekamy na linie $89.
        dc.l    $01800555      ;zmieniamy kolor.
        dc.l    $8a07fffe      ;czekamy na linie $8a.
        dc.l    $01800000      ;zmieniamy kolor.

        dc.l    $fffffffe      ;koniec Copper'a.

GfxName:
        dc.b    'graphics.library',0,0
```

Charty i Czarty

****

Niniejszym zapraszamy wszystkich do wzięcia udziału w tworzeniu **Amigowych Ogólnopolskich Charts'ów.** Już w następnym numerze zamieścimy pierwsze oficjalne wyniki. Poniżej podajemy kategorie, oraz zasady głosowania.

Kategorie:

1. Grupa, zagranica. **2.** Grupa, Polska. **3.** Demo, zagranica. **4.** Demo, Polska. **5.** Koder, zagranica. **6.** Koder, Polska. **7.** Muzyk, zagranica. **8.** Muzyk, Polska. **9.** Grafik, zagranica. **10.** Grafik, Polska. **11.** Magazyn dyskowy, zagranica. **12.** Magazyn dyskowy, Polska. **13.** Swapper, Polska. **14.** Lamer, Polska. **15.** Gra.

---

1. Głosować może każdy, kto ukończył 7 lat i umie pisać i czytać. **2.** Głosy zbiorowe mogą oddawać jedynie znane grupy z zaznaczeniem ilości członków biorących udział w zbieraniu wyników. Najlepiej jednak będzie, jeżeli do jednej koperty włożone zostaną wszystkie indywidualne głosy. **3.** Podajemy maksymalnie 5 propozycji (może być ich mniej, choć 5 jest optymalną liczbą) w każdej kategorii. Nie jest konieczne (ale wskazane) oddawanie głosów we wszyskich kategoriach. **4.** Punktacja jest następująca: za umieszczenie np. muzyka na pierwszej pozycji - 2 pkt., za pozostałe miejsca 1 pkt.

Gwarantujemy sumienność i uczciwość przy podliczaniu. Propozycje należy przysyłać na adres redakcji Kebaba, KONIECZNIE z dopiskiem **"Chartsy".**

```
0 REM ******************
1 REM * BASIC-STARTER 64
2 REM * (C) POLONUS
3 REM ******************
4 :
5 OPEN1,8,1,"BAS.START.CODE,P":PRINT#1,CHR$(0);CHR$(192);
10 FOR T=49152 TO 49225:READ A
20 PRINT#1,CHR$(A);:B=B+A:NEXT T
30 IF B<>7478 THEN PRINT "ZLE DANE!":CLOSE 1:STOP
40 CLOSE 1
50 :
1000 DATA 120,169,0,141,0,8,32,191,227,32,163,253,32,21,253,32,24
1001 DATA 229,169,1,133,45,133,43,169,8,133,46,133,44,160,0,177,45
1002 DATA 240,12,72,200,177,45,133,46,104,133,45,76,30,192,165,45,24
1003 DATA 105,2,133,45,144,2,230,46,169,0,132,55,169,160,133,56,88
1004 DATA 32,89,166,76,174,167

READY.
```

```
;——————————————
;start dla BASIC po dekompresji (C64)
;(c) polonus/commodore kebab
;——————————————

*= $c000                             ; start assemblacji
c000 78                 sei
c001 a9 00              lda #$00     ;
c003 8d 00 08           sta $0800    ;zerowanie $0800
c006 20 bf e3           jsr $e3bf
c009 20 a3 fd           jsr $fda3    ;system reset
c00c 20 15 fd           jsr $fd15
c00f 20 18 e5           jsr $e518
c012 a9 01              lda #$01
c014 85 2d              sta $2d      ;poczatek BASIC lo-byte
c016 85 2b              sta $2b
c018 a9 08              lda #$08
c01a 85 2e              sta $2e      ;poczatek BASIC hi-byte
c01c 85 2c              sta $2c
c01e a0 00     loop     ldy #$00
c020 b1 2d              lda ($2d),y
c022 f0 0c              beq exitloop ;ostatnia linia?
c024 48                 pha
c025 c8                 iny
c026 b1 2d              lda ($2d),y
c028 85 2e              sta $2e
c02a 68                 pla
c02b 85 2d              sta $2d
c02d 4c 1e c0           jmp loop     ;nastepna linia
c030 a5 2d     exitloop lda $2d
c032 18                 clc
c033 69 02              adc #$02
c035 85 2d              sta $2d
c037 90 02              bcc *+4      ;ustawienie
c039 e6 2e              inc $2e      ;wektorow
c03b a9 00              lda #$00
c03d 84 37              sty $37
c03f a9 a0              lda #$a0
c041 85 38              sta $38
c043 58                 cli
c044 20 59 a6           jsr $a659    ;RUN
c047 4c ae a7           jmp $a7ae
```



LISTY DO I OD REDAKCJI

Pascal z wkładką?!

Drogi Kebabie (10 tys. zł), postanowiłem napisać do Was, ponieważ znalazłem błąd w HighSpeed Pascalu, a konkretnie w jego bibliotece. Każdy, kto choć trochę bawił się Pascalem wie, że poniższy program kompiluje się bez przeszkód:

```
program qwerty;
var
f : byte;
begin
f:=255;
end.
```

Teraz dorzućmy 2 linie.

```
program qwerty;
uses
exec;
var
f : byte;
begin
f:=255;
end.
```

Kompilator pokaże błąd - "wartość zmiennej poza zakresem". Jak świat światem zmiennej typu "byte" możemy przypisać wartości od 0 do 255. Co jest przyczyną błędu? Otóż w module "exec" znajduje się zupełnie wariacka deklaracja typu:

BYTE = ShortInt;

Aby przywrócić prawidłowy typ "byte" należy na początku programu, w sekcji deklaracji typów umieszczać linię:

byte = 0..255;

Rozwiązuje to nasz problem; powstają jednak wątpliwości co do solidności produktu firmy HiSoft. W tym momencie pragnę zaznaczyć, że błąd jest w module, a nie w kompilatorze. Mam nadzieję, że nie będę zmuszony do szukania kolejnych błędów.

Z coderskim pozdrowieniem: MACiAS / Lunatic Asylum.

Dziękujemy za powyższe uwagi, i zastanówmy się co może być przyczyną tej "wariackiej" deklaracji. Otóż jak sam zauważyłeś i pokazałeś w pierwszym z przytoczonych listingów, kompilator poprawnie rozpoznaje typ "byte" i pozwala mu przyjmować wartości z zakresu od 0 do 255, co jest zgodne zarówno ze standardem samego języka, jak i jego borlandowską mutacją.

Co jest zatem przyczyną błędu? Dla nikogo nie jest tajemnicą iż moduły Exec, Intuition, Graphics,... itp. są ściśle związane z unikalnym amigowskim systemem operacyjnym, a ściślej z bibliotekami występującymi pod tymi właśnie nazwami.

Firma HiSoft, aby umożliwić łatwe korzystanie z procedur w nich zawartych postanowiła zaimplementować moduły będące w zamierzeniu interfejsem między kompilatorem Pascala i samym systemem. I tutaj zapewne powstał problem stworzenia sensownego i logicznego systemu oznaczeń stałych, charakterystycznych rekordów (czyli struktur danych), a także typów. Być może dlatego postanowiono uciec się do sprawdzonego modelu - języka C, gdzie typ BYTE przyjmuje wartości z zakresu od -128 do 127.

Ale jak zatem wytłumaczyć kolejną definicję znajdującą się w module Exec parę linii niżej, gdzie typowi UBYTE (Unsigned Byte) przypisuje się identyczny przedział dopuszczalnych wartości, zamiast zakresu od 0 do 255? Niestety nie mamy odpowiedzi na to pytanie, jednakże jeśli ktoś z Czytelników ma jakiekolwiek propozycje rozwiązujące nasz problem to prosimy o kontakt.

"DATA-TOSTER "

```
$0801-$21B0
0801: 2Myg 1FU! c3gV 0a00 ue&1 KgMx 68
0813: CvE0 !d3T j001 K3jZ Px1P 16Mt d6
0825: !LtG Dz3C Tfyw JiGe tJPc v5hs a1
0837: X7p6 %5hy WnEN gIwM dxPC 0mfk 68
0849: 1veL c4F# vCPM f3&Y 45UG CF4# 32
085b: &rGU n3NI 031Z 31EM v31y ynRh 78
086d: 04z9 #PMs 7sMg c7Z% c10h BBQU f0
087f: cp9f gUL1 gL#0 TNxA G2Ru &P%1 9f
0891: cCNb f3ND pzIH kMrj 0Fwc !Jne f2
08a3: kZ44 EWT% jX1# G37A Aj3J HNiG 45
08b5: Vxys KB9w nDqn rDpU J3xU x10U c4
08c7: 7dg% iHXx PM&e 7zp% 1Iai Kjc7 26
08d9: Wkfe UhLe Nb60 Zq0g A7LA fEKv de
08eb: 90tc wN&5 xgVM ffc& YMJa wK0c ac
08fd: sfc# YNxM cPU6 mcu1 CpVG TF6H b3
090f: #fFY Sh%A 0W2# &&3M UgVC ZvXn 93
0921: zVVB &0vH 7mr2 %A50 sEQu 1wMM 9b
0933: Qisx &5T1 P&3m %L!R XLXm KgNv 80
0945: tDp# rCXH xPNx 7C1w x0Ym F6pI be
0957: dFiR ZDNU r9Bn 9iC# X7Vq 0gbM 11
0969: E3wk 0e&z xAwE 00v6 RLXK NJ&3 75
097b: 81kE 00IE 0sEg V!aW hUhw c31V ad
098d: tisB 9kAt 6bg6 3133 073r 3AMh 54
099f: GJEs sg2p c&JF Rw30 Yt&Z vC&c 72
09b1: vz2J nw38 55w5 1jcP Y61I u71C 07
09c3: RxVB %e41 pDMs &00i nYG5 N5X% bc
09d5: cfY# &69Y VgBK 3w&# pzY6 R4L7 6a
09e7: vC0# 33N2 Cq6x Ck9S zg0M i7x8 89
09f9: i6ar 0C30 0#2! xNM& ebDF fM00 c5
0a0b: awcX 3LPC 0c1f f1zW pKsn 61j! 5e
0a1d: QIf! 51yV Qhwg f6pw pzM0 30x# 4e
0a2f: j1w! vKhX a0I2 Y600 3Mc! tRwP 54
0a41: VI%1 2c!s iIAN Ny2U 2Yrv I%a5 a9
0a53: AsBQ x#7! #A&u b9%w 55ui iHCm f3
0a65: IufX aYsv Ep7P 1R!q vKua egaS c8
0a77: 3K1X c&Rq DHgz 7Hhm N&Az cUeh 29
0a89: yIIj y&yv iyNt bQ&8 PYDz SjNw 6f
0a9b: zbFw %x16 2&61 AVU7 !sf2 iQkP 8e
0aad: Qytc 5Vub MMTf E2%P Vs#3 PVSN 68
0abf: bME0 YTQx MWe3 eu%f w83f XYa! 41
0ad1: 2!t6 YCBf &Ai8 ww3v b87q UFPR 99
0ae3: k!T3 JIf3 C9D4 J39B 3Fs0 3vZ& c1
0af5: N82q &bg0 ZqAm wnn0 &xXf Jr%i e9
0b07: G1FQ 9s3g f#Uf A&DM yry7 #B0U c9
0b19: UZg% iHXx P%DN UsC0 #sai Kjc7 27
0b2b: Wkfe UhLe Nb60 Zq0g A7LA M&Kv 63
0b3d: 90tc wN&5 xv5M fffD YMJa wK0c cd
0b4f: z%c# Y#uf cPXV mcu1 CpVG TF6H 77
0b61: 1%G3 Sh%A 0W2# &&03 #82p %g7R 22
0b73: ESuv Ss5W xVCM vN2g ne6z 0vdc f2
0b85: Z4A8 m5tM cW0R 0nYh 0iBJ bIe9 91
0b97: y&6h AtvW 8k#& 1V#v U&c5 apCj 6a
0ba9: E&0b wUuj AHMG iv5B 3U6B k4V! 8e
0bbb: !1z% 6j54 gLYV eiA1 4jCM 7w2F 89
0bcd: gfZp g0C4 V1K8 Xx6v PYZx nEH& 7b
0bdf: SJFi 1#sJ #vf4 cf#f 9f5j 9QNg ea
0bf1: Dn40 Cj4b w82& 0miY tk#1 Cve1 ab
0c03: PWIn 0f!! fPcP PcMf DUh5 46zL 29
0c15: XTza I%%7 1e3w Y0Em 0L3D %!3% 62
0c27: &vX1 DYm3 !w6& HcfV MpD0 #s8u de
0c39: 7&6v Mvdb CgGw VMs7 4KsU 83Yr 75
0c4b: A1YZ U#v7 P4Q7 0fZG 0e2j Y#v2 c5
0c5d: CsYL %LPV AUuf DSg1 p5bD XYep 77
0c6f: DVD3 %%fT wrfD P&7! fhg0 %Mcf 9f
0c81: 6f2G y4EX 0ftJ b22v !v#3 SaB% 0b
0c93: zg3s HzqF dq2v f1qw &Rhf YkUY d0
0ca5: LaeE mI4V q61x WsF% WkmK wQ15 97
0cb7: BQVM pDpF R1JD swAe KB1f cKj# fe
0cc9: ii1n !q&Y QInk Rtbe bw3j vyni 53
0cdb: ianj iaMy QWD0 iwqF 1Q1d azJd b2
0ced: 4f06 t1XE QfkF QRPi q8xL FDzz 2b
0cff: KW2b 87cg AIGc Bmhb MHU2 mjwK 82
0d11: pMr4 f%D0 hgze D11c NN3H zcEH a9
0d23: qksw IfcB CjAJ Q0IE DsA2 Aej6 0e
0d35: 0A!Y 46Xr SW21 3p7h AcjX d#N5 5a
0d47: 722t M0tk 76F4 b#rI 7zWZ #xU& 2d
0d59: rrfK VLMc HySJ 9ADX Hish xmaR e7
0d6b: ce#B #Wg9 SAnX p#XY Da1t 813X a1
0d7d: b150 FvIY Lvdw cjeW Q4!H %W8D 9e
0d8f: GtZB 2nEg #C3a 06d9 w30u &4z5 39
0da1: Ft4& pte5 Qp01 !aFE Fda5 %UrX 63
0db3: xf!E iLPq !!PU fvW5 NA9S w9xz 33
0dc5: Sxz# W0bc Tfb2 EUU! 0Ewg Z5dH 41
0dd7: 0C39 7t0e x7R6 IdnU w979 Dt0c 27
0de9: hgsT Nj%6 S4r9 5d0D EBLM Jt48 9b
0dfb: m7YQ 5ExD !dWM 1cxc ZK4q &cYV 5f
0e0d: FKqM JVjg c21Q Rxb5 Dg4r L8y4 ab
0e1f: %qj% ycjZ kaSP H7yp wgeN #YyM ae
0e31: B8xc axbk Gi1p 98pI nsAw A3j9 e8
0e43: gf0M gTLM bcAy Y2z4 fyv9 If0z 7f
0e55: !rPM 7YCN Y1L9 JL0n !qHM 4YCV 18
0e67: Y0%9 HL0b !qTM 1YCZ &uWX wqwh f4
0e79: FfWp kTZP &907 atYU Wra2 eyA% 8f
0e8b: AvL4 %%02 VLVc Fh5U Ggyd Z2Sd 46
0e9d: 0tiF Y8Q6 Ra8w !xTE Ya00 Ew28 93
0eaf: QfS3 beSe 6die 1dh& &8XS giNj 3f
0ec1: 3rcs &205 AMyi n4Ri stq! cjCt bc
0ed3: 7AbS JZ3f Pc%e RteU YNrn RzcK 9b
0ee5: c&IL cP0K c3AK ejb4 HqQK Q19a 34
0ef7: hr&r !Q#S z6tj iqHe NEmC Nsb1 9b
0f09: MJNn 1s7F a2Ew MkNc xhF9 hQxk 60
0f1b: qpRi h1dD D5rf 1M25 V&63 fYhf 9b
0f2d: jQKh gB6D I5kA 1mDU K18a pAQn fe
0f3f: WjqJ Ws9p pL&w PgKJ 1jEw 3qk0 81
0f51: TK!x cm%f h0y5 ugw& kDpB LCA! a3
0f63: vYd5 Nxpv t2vN fV11 !mLC cTgz 68
0f75: SxHh jWT& bxg# DhCb apad deUx 2f
0f87: VdaB cj8D R1NE ij1W 4d95 gAZf 2d
0f99: 1661 WB4Q !Zb4 H!wJ R4#V WtMd a0
0fab: SP&F NnCs 9G0q aM59 t2Rb xMQ! 3e
0fbd: 8dtp c2Cd ncBf k4eC zQJc &3fI a8
0fcf: hvBR B!0N 5akS Qhhc GtRr 9A11 4d
0fe1: AsKR sJWd 9XJ0 kBgJ R10D I41n e4
0ff3: 75bB HNm5 CtU1 82V1 1n7c LySY aa
1005: a1He dm91 z#PM I23L V74n h3vR 09
1017: nx!M 2%Hn gsnb EbFz advQ 82t2 2a
1029: bysF &8Qd ikX8 pwB1 iH6r Cti1 34
103b: 1&Q9 ntfI 58Nj xVhE hUbb jUcV 8a
104d: R21e Kyje hak7 &VNq zGRF Cc1x c7
105f: 1bBn 3teQ 28e4 ii%g B1ut 4RRq 8f
1071: gABr pFbe ZGme bc48 DQrE p4S7 49
```

```
1083: uYbe QzHe PR4y hjTj Rc7i MsX1 c2
1095: SJv1 tidq jkDA gutf eBkd RFqr 38
10a7: kBGM h4KG M7Z6 qXF9 h1hb igRv 2f
10b9: 8d22 kfCj W0Qd QAZq iQ5q ew1c 3f
10cb: 92aL yxpc YV2a yThh SSce uUnc 1b
10dd: 30JX Ggiy 1G0n xvG6 #8jV L8WC 76
10ef: #GBn FGLZ 1xeN #8e& 6atZ YDTq 07
1101: lerV VLGB #IA5 Qdn0 Ed3h BTMP b8
1113: uTbe X3ts A9wR &7tr 6WmM 2D82 91
1125: FeS5 Fg0T uMqB Fd02 NF0w Gxpw 8f
1137: Gh7w 2Zun nhU3 TNdG Dzyb 44WY 52
1149: jjJw iclT 80N7 r!2V &x9d 9U3q de
115b: 3mxU 030T yAPt Xqiw v37C !j9f 88
116d: Pbpw Hp19 28ys 7a8G jGJd 1cBi 3a
117f: Y0ra QeFc rtJ# 0%x6 yd3T G0x& cc
1191: 5u8v GbD9 0iwg 0wAg NgDg T6%f 35
11a3: 733X D0yU 7D0d 0#Q6 F53# KcQe 14
11b5: 79A0 0qv0 Z8gM x3iF 0SqV xj&w 5f
11c7: VLtm xvV9 x826 2q1j Y0ay %U&S 81
11d9: !EFu bW11 0jP2 dccm 1Uli !9hS 0e
11eb: 0&kN EbGD qx90 kKIp kzGz iCQc 6d
11fd: 6li5 kN5i FBe1 mpwC 3Ulk QE11 44
120f: 2ypk gHSk Fapk 7qC9 g!U9 hB1G 00
1221: i0#G Lq3U ib4M ah#G q1T0 #20g 70
1233: 1sxc A0i4 %WzX 9qG& yw1a G52m ca
1245: 2GYd #rSW AI5r rbBb 1lAa agYw 43
1257: !TyA %Y&S W2HM 4m0f 1MQ5 2Mc9 a5
1269: 0gU6 30ga 0ww0 q6wf vHY9 Nhvc 42
127b: 2d06 j004 tu!U &DZY %WD8 xmge 8c
128d: P&Q7 7ak& xgyB 6&k9 tDug 0qk1 a8
129f: 9MtV 0t3Q j9j1 93F2 byHb glh1 55
12b1: j4Z7 WwJ2 nhzy jpbn gQCK jABa 0a
12c3: 8dfg Msf5 35YK Ay0i 1vCy P0nt 4e
12d5: KgWb V!9w Dqih &mg4 hU2x &2BL 3e
12e7: ExWJ 0tMg 6ggX GC05 4syp 0C4y 29
12f9: QfzM PNh2 HG0& !AYv 4%bV &41U 98
130b: SabX CGAJ MXk3 GhCd 6gey 8aEd 70
131d: dM94 Dg3k DaPP %Iaz mxnZ yfTB aa
132f: u22% UWC0 z&E2 Ghad 61Xv 8!St 6d
1341: zi7g 4Apq riNc Iw0f sQia NCWI 6f
1353: 8G&D RUM6 9Y%% dSVe BQpG ycey 19
1365: 0xl# UQh5 !jte 9HnY jhvg dSx5 31
1377: E78r 5GQl 4qP! VUmI x4Bb G1pK 85
1389: WDzC 0p5k hZLY Fp3M Xi0x 5GuJ d9
139b: EJ3w e1fe u21j EnzB 5&Pg t66y 02
13ad: AG0j J9gg dFLI dd0d IrNA 8ceE f5
13bf: qyMR YyW6 Hj7g aWqj xr&j ebAq bb
13d1: 3jJn RNba 4ftj JsmL 6sA! Q7c& 32
13e3: RA2N E1gJ 0vrh Gi6m 0883 9Ayt e9
13f5: #2Xb J%z6 2bq! Ewyw 0mz8 KEYa b3
1407: EIKp Jd85 !mVM z44h 0WW4 si2k 13
1419: b9cj !jfg psAS 5QX# chFw ZQYq e0
142b: v#nC Me60 Qp9L eqk% 6etd Pxam 0d
143d: d%&L rWB8 c0DG 2BZb FvXM MqTR a5
144f: 6I4w YbH9 nRfT u2gW SB1x Ph9r 1a
1461: E1le 3CYw P1zi !1fM 3cBm Y0w9 7b
1473: HqLM 12fk XsY1 mVPc n1&S VfWx 59
1485: R4M8 6WU4 073R CUza zDh# zfVU 80
1497: Gjyp 1Bqx ISAw Hrhy pHkd E&LM 80
14a9: 8Kbh 1GTT 4j6u pFRS pH%n 02SJ c9
14bb: WGgy HuIr xsgq &%ky iwGd Cyd1 14
14cd: Bi2h 8AIE tMp3 pqwR cwQY Y20k 16
14df: 5iak b#&w zgkx Vybi v2eJ wQtj 8e
14f1: E1Le DNJc dN!F dUk1 m60j MI7q 65
1503: iDT7 X7IN Zchd vsyz MGw4 xbxa 7f
1515: Iq3d 6RYw RR0Z 6zX1 V%Bg YD1W e6
1527: B4wL VA%d pIE9 zcT5 jtDA gcS5 e6
1539: hX43 H8yQ kKVJ 8KUJ Na7w Q6yw c7
154b: KGII iaFe x9pc y1hP mLk! Hhs1 43
155d: ndeD &!pf mjrW %AmZ cew6 0NtD a4
156f: ve6w xdG1 61tX 52v7 xA!B bX!k eb
1581: 3H8V xz9A tGzx &dhw EKLT 11Bw 12
1593: crGS 2p0w 3sxU KA8B skMQ 7GBB f0
15a5: C3EF vPTx 51n9 UD9J #&4p MaBW c6
15b7: E1Mq Lp1z L1P8 zGDc 61G1 FhNI 2a
15c9: 6RIA G5wl 127n IFZD SxnA GrEI d7
15db: qgTK E1Rb q6TD YBnW PcJE rGCi 2d
15ed: 3xK& sH4V GHpm ek1b 8SCE n9dL 29
15ff: 4vNF erSq 52GM YI1I TLrv 7Q!L 6e
1611: juT0 U8zs g620 Ea&0 00&1 Rd%q 46
1623: LEqS gh5# &UUQ hhcB X591 ye33 33
1635: r&d3 D7j2 1m&1 &1h3 h4Ma SYbt fb
1647: kNcd R5Bk 1rr& G79p q!6K hXAL b5
1659: 132N DAJc 1n9p eyC# Km&p e!IL 29
166b: 8QEB z4jD z1hU kAZ2 fdQR CsV4 35
167d: 84EF Dr2t i6lz Nr4J skhq aPVE 31
168f: 64ti mjfi 1Fze g1Gp Z703 6#fw b0
16a1: j0sU PkJ1 Hkgc 1s%3 8cQd Pk5b 8c
16b3: kRBd 8vLj MQN9 yet2 TyFD Klv8 af
16c5: BMjF izCa ux2c c6Wd 0rYK WgVC 89
16d7: yUgH Yz2a jyxz FQTL yI!Y Gnf8 4c
16e9: 303X QXuQ 0yUJ 7YCM !mVM Bib5 74
16fb: Ci0j 4gCl jEwg #22v 4ar1 DqWJ 43
170d: Ierl 28ez !gGg Lkyd di8w 6ESF a5
171f: GnYl XhmH vh7M YwTZ RpSR BET! 40
1731: xnsh MLCg s2Y& ps5k brz2 #Xas f1
1743: u3#E zksl znix Rz6J !pT1 HsEw a8
1755: h7G9 6EGd vGn2 56eQ MH0p Fs7G 12
1767: yY6g 1er2 Y0LK BF&F E0!y jdQw c8
1779: GsbR wn6K Z9fS 0e2w 7!c% &y#K 9c
178b: &ira HMAN NKjZ 7PTa aAU! 139D 9c
179d: DShi C4Nv 8gR3 fQZi mHdb udWN f5
17af: PLt5 mAr0 htZL GCvh J%y0 XFXP 05
17c1: 3EXD 8q#1 Mr!7 QfKF &AhF kZNa 40
17d3: Cyt! aAYg jb4X 69dd kFN0 eBI8 f4
17e5: TpWN waAZ gMqF gJyR &ZkX G3Kz 34
17f7: Q%kw 4shR iHjm hCYw BIW4 Fv%9 2d
1809: jS!n G%ZL xcQw EhKN 95yB 8nku 71
181b: p3Yx Fibk rmcL 9R0D e&by SG&w 57
182d: F24R GK18 cnYw Hj8y xi4y tIxc fe
183f: Q28U FqXB H5GG Fq%B HsXW UfZF a0
1851: ncya NGjg 1GD# xqlI 2QzY zx3X b1
1863: 84G0 dh5Y 1a&1 cCxK qqGZ Cy9j 15
1875: 0QFa av3Q PI1i 0TPt &020 8a10 e0
1887: M63w 490M I53g sf19 Gg6y GG0y 70
1899: qvVi WWAf U5e9 8w1y bYs0 VAhB 17
18ab: &8qK xa#G Jpel Hbk1 xqRM 1h7g db
18bd: xpQi 6LW5 KqA8 xs!5 Ky31 YRSI 39
18cf: GlvE Gi2G Gi!V hycw TuT8 A1SP dd
18e1: 1He& QeeF hq2m Ewns Fx&w %KRU 65
18f3: KSbt cfIU HqhN k901 IqP6 5d5w 28
1905: 8KqI Q0b& XsqB QeSB Ff3C 84bS 89
1917: jaIy HpvL zkFF 0Uml xj6a qgq5 fb
```

```
1929: cDyx Y0!5 wqGE 5LYu a&3M wYbY 48
193b: fMGJ #wbg Ij9! j6DV fFTN Zau4 8a
194d: Bcqz 2f2h Bcww &wmh ccz4 %Z3S d0
195f: xe#r 0uqS HJdT F&6l 1Wm0 RrfL d5
1971: IVk6 j1zQ &FcQ yaA4 b7Ck #140 2d
1983: WLDG Wy1Y 081w M02J IMF8 q0R8 aa
1995: 8!Af 1s18 Ggad JktE &a9A LnjR cd
19a7: IYCF &8SQ 0&r% W8q2 xEcw BtYM e8
19b9: #vE0 68RX dwGd %wmG Gk2d #gaR 60
19cb: 1L0r xgGF U8k2 Fg8M %cA2 AeX9 51
19dd: swTT ztn8 MqX% 1kMa VG89 LuQ5 08
19ef: DkY1 !J3T j501 8anr 8fjK j2vi 85
1a01: 08PX 7Kh1 gMNr ssPb Aaw6 MBuM d1
1a13: YhK# kBT9 HHMw cZV2 BZGv 5izk 87
1a25: ZZK7 mR9I SPXu 1DU1 8WP5 vjc! fd
1a37: jDWh #MIL d#eY jdw8 !TXj D682 56
1a49: sCqi zj7z IlE2 9M2c ariU DsEx e2
1a5b: nI81 9EXb n8yc ZWbc tMnI g216 c3
1a6d: rJZ2 z7iy 7110 UqTW Enw! 1xdf 32
1a7f: 9qUC PaNY kZki HcMR imLK QL10 98
1a91: Q!jK R2gl idkA j1gB GgFM aire c9
1aa3: 5Vb4 Kduw PPzn &2!Z 6as! UjGS 8d
1ab5: IU8c MkEn gRGM 144j c21c ilh5 18
1ac7: kC6J TJgr nkCq gA7W SKz7 iQBy 66
1ad9: YkR9 iF93 C3YD h75b nPED ayuc ca
1aeb: cJtq Kl8w ewcH LJtc Yk5T NrVn 6b
1afd: mAZi htIn h6xT 1QBq iube j!U0 17
1b0f: QQnn 9AJk mltj Mwki hQ!M pra6 69
1b21: T09I KlsK aQku MY8d P4d& 3kzl be
1b33: CJzg #qVi Kliu YPGP &&#P Bh!Y 7b
1b45: pj!c DUy& 87Qh mjeZ NR4G Y0&! 66
1b57: W3Y7 YUUS CFG0 C4F0 WdU7 0MmF 23
1b69: 0k!S 9wBb jYdB Vzqn TGR0 mu9p 46
1b7b: TY&v xeBX 11Gy g4MK 0&RF WG0B 5b
1b8d: JSjq keRk &d#T ybAY kiPu KdYD 3c
1b9f: jbIC !kPg TC#P cBYy iT0F &!1F ec
1bb1: bGmh jxSw 9i1Q i8fb whBs Lmyd 7d
1bc3: 14il 58r1 8WhI Ggm5 QWr1 &bQ! b6
1bd5: #S%8 CdT8 MV3Q NTv9 0%3A MGJ# 34
1be7: 9asD VI5e 5Wn1 Ppl8 AcKF 3kV7 c2
1bf9: 5ti8 IGBB w7Ip j5AC e8RM EV2K c2
1c0b: IjDg 0F6w 0bC0 0YA% Y0jh #Z08 0b
1c1d: !cM# 0V3L 660U &204 4be% zsqy de
1c2f: 0AK# pDxF aian ixlv aCr2 V&XY 93
1c41: AEPM G1EC STHu giEn gSHg 4e7! 3c
1c53: EDyl KUVQ zf4z g0T7 NiuM R61F 5d
1c65: CVI8 05IH n&1S kRiN k4Vp 65Jr ae
1c77: b1Sp M3qp R!Zq lslN 7gRj jZbF d1
1c89: Clfd jU2n hACE Bjbq zGKP 40Uh 08
1c9b: FIVv ZhiT 4B3R fqMx N4XZ K1an e0
1cad: wRkI 3kLf jkle J49u jPnV eZdb bf
1cbf: gipa b231 f7QS gVzz n%F3 mufL ad
1cd1: 3sVz 2ErU iQ%z jKqJ NWZv %4tf 22
1ce3: %iNl tujr nRBj i!4d MQy! Qs%L e0
1cf5: Hk7# r5dl L20E R8Z5 ajYw 035B 06
1d07: kMQC k%z& J#&k jETL HL3C HbWJ 63
1d19: f&ka NL9S 6vQl Nej% YfL9 aHcf 2f
1d2b: FRfE X62i XKV& t62h Q0KJ Bf3v f2
1d3d: PFzw hGb9 mC02 Wple U7iB 760W 01
1d4f: aZ0r iw#y 5nck Hr7K x1HK eBa7 cd
1d61: bt0q Hizg 1!&q kf3z Hu&Y qIXW 17
1d73: AsXL V4Mc i4Lg 7WQ& a1mr 39WF a6
1d85: Gq0E J7sw 7P64 1fGv 83Ix j20q e2
1d97: !kRG 4TEs qlf2 2t&A jetE 1Z06 72
1da9: 83EC 95r9 kwcq j!H9 ha04 byP9 ae
1dbb: jQb1 j18s hgwr z7Yb 06ry 8icA 64
1dcd: 9i&D a2AG a!MJ byYM cj8P d3kS 33
1ddf: dPwV ezIY fjV1 I493 L4h5 Ikp7 86
1df1: i4Ba iQ!S jkWG jXBg kuZZ GqXY 58
1e03: glpn m5Bq LqRr n5Ru n%vS 18n# c1
1e15: ul&B !z0f FvQ& vpD5 xvSg 0Kr# 76
1e27: !x3N &aSS #Z3K %3H9 0hv7 j0yc bc
1e39: B0KD ynbB P!kA xgCx a&VI 5&M% 1f
1e4b: 8HVx W2gD QbRM bUQQ Q#wG Hk19 a9
1e5d: 229W y2zc wUPu zL!K sM2I r!ZB 44
1e6f: 0sy6 M&j2 IiL5 MF0a gzT5 Mp03 3d
1e81: jd&G j5EH x9Kk kr2A Qw24 F84w 92
1e93: i1F1 qeNc UqWx Hdif !A0t 15QJ c8
1ea5: D0nt SiVt 7s6I ZCQw d1!J asRC ca
1eb7: bV08 HivS PmkL I0LK VyDg 0#XD ba
1ec9: WbRn aWA6 zi0m 6M2d q9Ch 6UQh 1a
1edb: Qfjl zmsL Gglc Z!xc sd!J B203 2b
1eed: 722e aV0t HmFb zuEF Hpg6 zuIF ed
1eff: jic0 bs44 Lg0K Dg05 &LrN HKwF 1f
1f11: HeAF jaEG Hwcg 82kh WTPr IvSc f5
1f23: 1G!N #!2P aUmJ FqP5 Hv01 &czc 85
1f35: X2Cg Vhxw QG#3 TuQF Y0fE Qfya 3a
1f47: &0Tb LkQF 8iGj kRF3 CpYw eQDh e0
1f59: jxRa h&50 Q0Tg jR1i mAl4 qzQp 80
1f6b: 1Qli kQF1 84iR A1B3 u5Ff kRgJ 36
1f7d: fva4 hR91 jA54 j!1g gsAZ Ikd9 cc
1f8f: 85sw jkBG ald3 hil1 iRhl gkNe 23
1fa1: hkEK 0aCj 8dbJ S%wt Gi6d RiRm 38
1fb3: 0bQw bVTn 4Kzw 3F3R 87sI Fp18 ff
1fc5: 89gI qf0i GrZF v!ey 4G90 84kh 10
1fd7: jfEE Gumw a!0u G!0& 4cBe YeZc 8b
1fe9: HNCy 21yU HW00 8bFg Kx2y Rq0i 82
1ffb: 8bT% 8c3% EKAZ NDXf %WA1 8cf% c1
200d: jcP% Vcz8 !aA0 xh2C #P0d 1LEG b0
201f: 9x36 #Uzg YGE& &4yx %UnW Ewu6 57
2031: ##j% FLXg 0Ir% NLVE AdTw VZ3p 42
2043: Gju5 01xc 012F 4e&h 9165 1aGY 9b
2055: S0sw 90uC 17SW 1QyB 47T7 1WxE 31
2067: Fwbg 2c00 Q0j9 fv3& 66nY GFxB 88
2079: %&19 F17M 88EU Vh6M 0Yp9 e818 bc
208b: FvPB 4r02 NLS5 %b58 y97Y Nw7K 50
209d: 8d3C 0pzg Ym0E 1MMe ag08 63xV c3
20af: %7NY 010w g800 0000 0001 0w00 36
20c1: 0000 0gc7 3Mc4 1g&7 1Mw8 10g5 ac
20d3: 1ws9 2wIc ZXCV MSnc c3Uh cPy1 8d
20e5: ACec gn0n zbZD VgIz dKBU MKnR 4c
20f7: zjN& fI4F Hyil syRz 7ftn ee12 08
2109: 8#uL Kg6s b10w KgMw ChM7 !d3T 2f
211b: 82c7 Y4&w 8Mvg c20y 1SA2 !gig 15
212d: 9Z07 82c7 qgjg 7y0x 1SA6 !gTg 6a
213f: 4sww 8gtF 3sAk Q0uw 1i0A 1SAt 52
2151: Xw04 WEkh FLWB %!2a 1W19 xv#B a9
2163: i8n# 82c7 xgbM 50EI Gge5 4i0x 9e
2175: 1W&2 Q09F 221x 1%2q 82c7 YeAw ca
2187: 8wtF 1cA6 Aebg 1!0y 1SA6 QdCw de
2199: 1i0A 1SAa !hnM 3sAm QcGw 0i0A 48
21ab: 1SAV Qc7P t!Uw R6dF pvJH &i0a 51
```

```
/**********************************
*  Program demonstrujacy          *
*  mozliwosci komunikowania sie   *
*  CygnusEdytor'a z PowerPacker'em *
*                                 *
* Autor: Krzysztof Kobus          *
* (c) Kebab 1992                  *
**********************************/

OPTIONS RESULTS

ADDRESS 'rexx_ced'                    /* Sterowanie do CED'a */

Okay1 'Przekazuje sterowanie do PowerPackera... Do zobaczenia!'

ADDRESS 'POWERPACKER'                 /* Sterowanie do PowerPacker'a */

PP2Front                              /* Ekran PP'a na wierzch */
DecrunchOnly                          /* Zdekompresuj plik o podanej nazwie */
IF rc = 1 THEN                        /* Czy plik zdecrunchowany? */
  DO                                  /* Tak!!! */
  Notify 'Plik zostal zdekompresowany.'  /* Wypisz komunikat */
  Save                                /* Zapisz na dyskietce */
  GetFullName                         /* Pobierz nazwe pod jaka zapisalismy */
  Sciezka = RESULT                    /* i podstaw pod zmienna Sciezka */
  Bool=1                              /* Ustaw znacznik */
  END
ELSE
  DO                                  /* Nie!!!! */
  Notify 'Plik nie zostal zdekompresowany, lub podany plik nie istnieje.'
  Bool=0                              /* Skasuj znacznik */
  END

Notify 'Przekazuje sterowanie do Cygnus Edytora...' /* Wypisz komunikat */

ADDRESS 'rexx_ced'                    /* Sterowanie do CED'a */

CedToFront                            /* Ekran PP'a na wierzch */
Okay1 'Witam!!! Jestem spowrotem w Cygnusie!'

IF Bool = 1 THEN                      /* Jezeli znacznik ustawiony */
  DO
  Open New                            /* Otworz nowe okno w edytorze */
  Open Sciezka                        /* i wczytaj zdekompres. plik */
  END
ELSE                                  /* W przeciwnym wypadku, */
  Okay1 'Power Packer nic nie zdekompresowal...'  /* wypisz komunikat. */
```

Od redakcji...

Na listy odpowiada Paweł Sołtysiński

Bardzo ciężko byłoby nam udzielić na łamach Kebaba odpowiedzi na wszystkie dręczące naszych Czytelników pytania. Jako szef działu Commodore 64 bardzo często otrzymuję wiele pytań dotyczących obsługi tego komputera (najczęściej przez telefon - około 20-30 telefonów dziennie). Oczywiście, wiele z nich dotyczy podobnych zagadnień, które nieustannie się powtarzają. Myślę, że taka forma grupowych odpowiedzi powinna być pomocną przynajmniej dla części naszych Czytelników. A więc, do dzieła! Do najczęściej powracających pytań należy:

"Mam cartridge'a Black Box i nie mogę uruchomić niektórych programów. Dlaczego?"

- To pytanie w dokładnie takiej formie nie występuje. Najczęściej jest to dialog typu: "Niektóre programy po uruchomieniu 'zawieszają się'. Co zrobić?", "- A jaki Pan/Pani posiada cartridge?", "- Black Box'a.", "- Nie mam więcej pytań.". O co tu chodzi? Otóż karta Black Box (różne wersje) jest naprawdę przedziwnym zbiorem kilkunastu UKRADZIONYCH programów, które z osobna są bardzo użyteczne, z tym, że ich twórcom raczej nie przyszło do głowy, iż ich programy będą zmuszone współpracować z innymi w tym samym czasie w tej samej pamięci tego samego komputera. Wniosek - aż strach pomyśleć, co dzieje się w pamięci. Osoba, która złożyła to wszystko

c. d. na stronie obok

Errata do poprzedniego numeru.

Szalejący chochlik dał się nam we znaki także w poprzednim numerze, nieznacznie modyfikując listing do artykułu "ŚWIAT DŹWIĘKU". Poprawki odnoszą sie do linii zawierających:

```
move.w    #SampleEnd-Sample,AUDxLEN(a6)
```

które należy zastąpić przez:

```
move.w    #[SampleEnd-Sample]/2,AUDxLEN(a6)
```

Analogicznie poprawiamy linię:

```
move.w    #ModEnd-Modulation,AUD0LEN(a6)
```

na:

```
move.w    #[ModEnd-Modulation]/2,AUD0LEN(a6)
```

Poprawki dotyczą również programów CODE-INPUTER oraz BASIC-PROTECTOR. Aby programy uruchamiały się poprawnie należy z obu listingów usunąć wszystkie podwójne znaki "<<" i ">>" zastępując je pojedyńczymi "<" i ">".

W następnym numerze znajdziecie oprócz naszych stałych działów także sporo innych ciekawostek m.in. Co to jest Packet Radio. Jak wykorzystać C-64 i Amigę do komunikacji przez PR. Podstawy komunikacji komputerowej oraz test modemu faxowego. W związku z nawałem kandydatów do "złotej klamki" planujemy również rozpoczęcie działu pt. "Czego nie powinni przeczytać Czytelnicy... (różnych czasopism). Oprócz tego jak zwykle sporo listingów, znowu pokaźny opis gry oraz opisy kilku poważniejszych programów. Odrobina humoru i wiele innych (mamy nadzieję) ciekawych spraw.

razem (dalej nazywać go będziemy Składaczem) wiedząc, co się święci, próbowała ratować całą sytuacje poprzez średnio zręczne procedury zarządzające.

Niestety, Składaczowi nie wyszło to najlepiej, czego efekty każdy widzi (w zamian za efekty pożądane, np. poprawne działanie Tape-Burgera). Jakie dalsze wnioski? O ile to możliwe, Black Box'a należy bezwłocznie sprezentować naszemu wrogowi a potem zakupić Cartridge a nie "czarne pudełko". Karta polecana: Action Replay.

Pytanie *"Jakie gry dyskowe można przenosić na taśmę?"* *również powraca bardzo często.*

Jest zrozumiałym, że posiadaczy magnetofonów konieczność obcowania z nagranymi na taśmie programami doprowadza do szału (osiąga on najwyższy punkt w momencie np. wizyty u znajomego - posiadacza stacji dysków). Generalnie odpowiedź jest raczej smutna - bardzo rzadko udaje się przenieść program dyskowy (tzn. taki, który nie jest zawarty tylko w jednym zbiorze) na taśmę tak, aby działał poprawnie. Stacja dysków została zrobiona po to, by zapewnić przede wszystkim swobodny dostęp do danych bez konieczności przewijania, tak jak w przypadku magnetofonu i naszej nieszczęsnej taśmy.

Wiele gier dyskowych tą niezaprzeczalną zaletę właśnie wykorzystuje, np. poprzez wybiórcze dogrywanie fragmentów grafiki czy innych danych. Dlatego też przeniesienie np. gry "MeanStreets" jest niemożliwe, gdyż praktycznie trzeba żonglować całymi dyskietkami. Czasami bywa możliwe przeniesienie programu, który po poprawnym załadowaniu nie będzie się już komunikować ze stacją (brr...) dysków.

Najłatwiej jest to sprawdzić w następujący sposób: obserwowany przez nas program powinien zostać załadowany a następnie uruchomiony. W momencie ukazania się czołówki programu należy wyłączyć stację dysków i obserwować, co będzie dalej. Jeżeli testowanie obserwowanego programu (najlepiej wszystkich jego opcji) nie spowoduje załamania się systemu na skutek braku komunikacji z dyskiem, to taki program nadaje się do przeniesienia na taśmę.

"Czy mogę prosić Silver Dream'a?"

- Niestety nie. Wiele osób kojarzy (zresztą logicznie), iż telefon redakcyjny (091-77674) znajduje się rzeczywiście w siedzibie redakcji, gdzie znajduje się również serwis komputerowy, prowadzony przez naszego redakcyjnego kolegę - Silver Dream'a. Stąd też mnóstwo osób dzwoni do mnie prosząc o podanie cen, wstępną diagnozę uszkodzenia lub po prostu chce z naszym Naczelnym zamienić parę słów, co jest raczej niewykonalne, ponieważ Dom z Telefonem leży w innej części miasta niż Dom z Redaktorem Naczelnym i Serwisem Jego. W punkcie serwisowym telefonu jeszcze nia ma, rozpalone nadzieją okoliczne ludziska kopią dołki pod kable a Państwowa Telekomunikacja dojrzewa do działania. Sprawa jest jednak w tzw. toku i być może się wyjaśni wraz z wiosennymi roztopami i topieniem Marzanny.

"Czy jesteśmy zainteresowani wydawaniem pisanych przez Czytelników programów?"

- Tak! Niestety, jak do tej pory nie otrzymaliśmy propozycji, która spełniała by nasze kryteria. Dowiadywały się o nich na ogół osoby, które w tej sprawie do nas dzwoniły, więc myśle, że poinformowanie o nich na łamach Kebaba powinno dać wszystkim jakieś o tym jakieś podejście:

1. Program, który miałby być przez nas wydany, powinien prezentować względnie wysoki poziom opracowania programowego (w Basic też można napisać solidny program). Chodzi tu o taki styl pisania, który niczego w programie nie pozostawi przypadkowi (mam tu na myśli jego idiotoodporność, tzn. właściwe reakcje programu na niewłaściwe działanie człowieka). Sposób podejścia programu też powinien być raczej poważny (z znaczeniu czegoś raczej poważniejszego niż programik do instalowania polskich znaków itp.). Tematyka DOWOLNA (byle nie obsceniczna).

2. Program może być napisany w dowolnym języku. Warunki:

(a) - będzie samodzielny, tzn. będzie go można uruchomić bez konieczności uprzedniego uruchamiania np. Simon's Basic itp.;

(b) - powinien być tak napisany, by w pełni działał na "gołym" komputerze, tzn. bez np. cartridge'a, sprzętowo-programowych przeróbek stacji dysków itp.

3. Program nie może być uprzednio wydany gdzieś indziej oraz nie może pojawić się na rynku jako Public Domain. Tak stało się właśnie ze skądinąd bardzo dobrym słownikiem Angielsko-Polskim (13 tys.haseł) autorstwa kolegi Krzysztofa Matuli - co za sens wydawać program, skoro już go mają wszyscy handlarze.

4. Mile widziane będą polskie litery, dokładne opisy w programie, duża prędkość działania, własna grafika i muzyka itd.

Myślę, że tyle ogólnych informacji powinno rozwiać pierwsze wątpliwości tych wszystkich, którzy chcą rozpocząć swoją przygodę i wejść do ekskluzywnego (jak my to nazywamy) Klubu Tych, Którym Udało Się Sprzedać Swój Program.

No cóż, to na razie tyle z najpilniejszych tematów.

Paweł Sołtysiński

Kupon ogłoszeniowy

imię i nazwisko

adres

treść:



Cruncher? czy nie Cruncher....

Aby dać Czytelnikom mały przegląd możliwości najbardziej uznanych cruncher'ów, postanowiliśmy wykonać pewien test. Otóż wzięliśmy pewien zbiór o długości początkowej 54886 bajtów i poddaliśmy kompresji kilkoma najważniejszymi kompresorami na Amidze i C-64. Zbiór do kompresji został tak dobrany, aby zawierał rozmaite rodzaje danych: kod w języku maszynowym, grafikę, sample, tekst. Każdy z cruncher'ów został ustawiony na swoje najlepsze możliwości. Ze zbiorów wynikowych usunięte zostały procedury dekompresji. W ten sposób pomiar efektywności kompresji wydaje się być bardziej obiektywny (wiadomo, że procedury na Amidze są zwykle dłuższe od tych na C-64). Rezultaty podajemy obok.

BLOK DANYCH :54886

POWERPACKER V4.0a - Amiga - 50848

TIMECRUNCHER V5.0 - C64 - 48827

LHA V 1.32 - Amiga - 47989

CRUNCH.MANIA - Amiga - 46040

CRUELCRUNCHER 2.5 - C64 **- 45393 !!!**

Czy potrzebny jest jeszcze jakiś komentarz...?